TAMPEREEN TEKNILLINEN YLIOPISTO

KARI HALME

MALLINNUSTYÖKALUYMPÄRISTÖN HALLINTATYÖKALU

Diplomityö

Tarkastajat: professori Kai Koskimies
yliassistentti Jari Peltonen
Tarkastaja ja aihe hyväksytty Tieto- ja sähkötekniikan tiedekuntaneuvoston kokouksessa 08.06.2011

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
HALME, KARI: Mallinnustyökaluympäristön hallintatyökalu
Diplomityö, 53 sivua, 1 liitesivu
Joulukuu 2011
Pääaine: Ohjelmistotuotanto
Tarkastajat: Professori Kai Koskimies ja yliassistentti Jari Peltonen
Avainsanat: Korkean tason tiedon hallinta, Tiedonhaku, Käyttöliittymä, Toimintojen keskittäminen, Työkaluympäristö, Mallinnus, Integrointi, Kategorisointi, Avainsanat

Monen käyttäjän mallinnusympäristöissä mallinnustietoa syntyy paljon ja tarve sen organisoinnille korkealla tasolla on merkittävä. Mallinnustietoa halutaan tarkastella eri käyttötilanteissa eri näkökulmista. Syitä erilaisten näkökulmien muodostamiselle voivat olla esimerkiksi yksittäisten käyttäjien henkilökohtaiset mieltymykset sekä tarpeet jakaa tietoa aikaväleihin tai organisaation rakenteen mukaisesti.

Tällaisen usein käytetyn, joustavuutta ja helppokäyttöisyyttä vaativan tiedon hallintaa tukevan toimintokokonaisuuden integroiminen suoraan jokaiseen mallinnustyökaluun ei ole kuitenkaan järkevää. Integrointiin liittyvä työtaakka ja ahtaat käyttöliittymät mallinnustyökaluissa nostavat esille tarpeen tällaisten toimintojen sijoittamiselle johonkin erilliseen, mutta helposti saatavilla olevaan paikkaan. Samalla voidaan kiinnittää huomiota muidenkin vastaavanlaisten, mallinnustyökalujen osana kehnosti toimivien toimintojen olemassaoloon. Erillisiin työkaluihin perustuvassa mallinnustyökaluympäristössä luonnollinen ratkaisu on luoda ympäristöön työkalu, jonne mallinnustiedon hallintaan liittyvät ja varsinaisiin mallinnustyökaluihin sopimattomat ominaisuudet integroidaan.

Tässä työssä esitellään Tampereen teknillisessä yliopistossa kehitettyyn Trinity-mallinnusympäristöön kehitetty laajennettava työkalu, joka tarjoaa helposti saatavilla olevan paikan keskitettäville toimintokokonaisuuksille. Tähän työkaluun suunnitellaan toimintokokonaisuus, joka tarjoaa käyttäjälle mahdollisuuden hallita korkean tason mallinnustietoa sekä organisoida ja hakea sitä joustavasti erilaisista käyttäjän määrittelemistä näkökulmista.

Esiteltyjen ongelmien ratkaisemiseksi ja määrittelyn tueksi esitellään joukko vaatimuksia, jotka rajaavat karkeasti lopulliseen ratkaisuun päätyviä ominaisuuksia. Työssä esitellään määrittely-, suunnittelu- ja toteutustasolla helposti laajennettava työkalu, joka tarjoaa laajennuksille yhtenäisen sivupalkkiin ja paneeleihin perustuvan käyttöliittymäkehyksen. Työkaluun esitellään laajennus, joka tarjoaa käyttöliittymän ympäristön hierarkkisten mallien ja näkymien hallintaan. Tiedon organisointia ja hakua, sekä näkökulmien muodostamista varten esitellään joustava kategorisointimekanismi ja käyttöliittymä sen hallintaa varten.

Työn tuloksena syntynyt mallinnusympäristön hallintatyökalu on integroitu Trinity-mallinnusympäristöön ja toimii osana ympäristön jokapäiväistä käyttöä. Se on helposti laajennettava ja tarjoaa intuitiivisen käyttöliittymän laajennusten hallintaan. Toteutettu korkean tason mallinnustiedon hallintaan erikoistunut laajennus toimii ilman suurempia ongelmia todellisessa käyttöympäristössä. Työn perusteella voidaan todeta, että esitellyt ratkaisut ovat valideja ja toimivat myös käytännössä.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
HALME, KARI: Management Application for a Modeling Tool Environment
Master of Science Thesis, 53 pages, 1 appendix page
December 2011
Major: Software engineering
Examiners: professor Kai Koskimies and assistant professor Jari Peltonen
Keywords: High-level information management, information retrieval, user interface, functionality centralization, tool environment, modeling, categorization, tags

Multiuser modeling environments produce a lot of modeling information that needs high level management. The user of a modeling environment must be able to organize the modeling information from a multitude of different viewpoints that vary from personal preferences to organizational structure and time based workflow management. Therefore a great amount of flexibility is required from the implementation and user interfaces that provide the information management functions.

Modeling information management needs occur frequently. Therefore the interfaces that provide those management functions should be readily available. Implementing these functionalities directly to every modeling tool would not be feasible, because of repetitive integration work required by the developers and screen space limitations in the user interfaces of the modeling tools. This raises the need for centralizing the functionality to a single location in the environment, outside the modeling tools. In a modeling environment that consists of discrete tools, a natural choice is to create a tool that provides an extensible base for common functionalities.

In this thesis, a management tool is presented that integrates into Trinity modeling tool environment, developed in Tampere University of Technology. The tool provides a readily available and extensible platform application for a developer to integrate and the user to access common functionality. As an extension to the aforementioned tool, a module is created that provides functionality for managing high level modeling information in a flexible manner via highly customizable viewpoints.

To solve the aforementioned problems, a list of requirements is presented that are used as a basis for defining the actual functionalities in the tool. An extensible base tool is presented in definition, design and implementation levels. The base tool provides an always visible sidebar and panels that provide a consistent and easy to use UI-framework for extensions. On top of the base tool, a module is presented that provides functionality for managing hierarchical high level modeling information, i.e. models and views in Trinity modeling tool environment. For finding and categorization of information, a flexible tag-based categorization mechanism is presented alongside a user interface for using that mechanism.

As a result, the aforementioned tool was produced for the Trinity modeling tool environment. The tool is integrated to the environment and is part of the daily usage alongside other tools. The base tool has proved to be easily extensible and provides an intuitive UI-framework for modules. The module that provides features for high-level modeling information management has proved to work adequately in real world situations. In the end, based on this thesis, one can say that our solutions for centralizing functionalities in a tool environment and managing high-level modeling information are valid and work in real world situations.

ESIPUHE

Haluan kiittää diplomityöni tarkastajia Kai Koskimiestä ja Jari Peltosta, sekä kollegoitani ja ystäviäni arvokkaista kommenteista diplomityöhöni. Haluan kiittää Jari Peltosta myös diplomityöni ohjauksesta ja kärsivällisyydestä kirjoitustyön eri vaiheissa. Lopuksi haluan kiittää perhettäni ja ystäviäni, jotka jaksoivat muistuttaa minua diplomityöstäni ja kannustaa sen läpiviennissä.

Tampere, 17. marraskuuta 2011

Kari Halme

SISÄLLYS

LYHENTEET JA TERMIT

MMI	Model Manipulation Interface. MMI on Trinity-työkaluympäristöön integroitu työkalu.
MUI	Management User Interface, hallintakäyttöliittymä.
ORM	Engl. “Object-relational mapping”, oliorelaatiomuunnos. Oliorelaatiomuunnoksen tarkoituksena on tarjota relaatiotietokannassa relaatiomuodossa oleva tieto ohjelmoijan kannalta helppokäyttöisessä oliomuodossa.
Trinity-projekti	Tampereen teknillisellä yliopistolla toimiva mallinnustyökaluympäristöä ja -työkaluja tutkiva ja kehittävä projekti.
UML	Unified Modeling Language. UML on Object Management Groupin määrittelemä yleiskäyttöinen mallinnuskieli.

1. JOHDANTO

Ohjelmistopohjaisessa mallinnustyössä tieto tallennetaan malleihin ja niiden sisältämää mallinnustietoa tarkastellaan näkymien avulla. Mallien ja näkymien voidaan ajatella olevan korkean tason mallinnustietoa, jota täytyy hallita.

Tiedostopohjaisissa tallennusratkaisuissa, kuten Magic Draw -mallinnustyökalussa [Mag11], yksi malli voidaan säilöä yhdessä tiedostossa ja mallin sisältö näkymineen esittää suoraan mallinnustyökalussa. Käyttäjä voi hallita mallitiedostoja käyttöjärjestelmän tarjoamien tiedostonhallintaominaisuuksien avulla. Keskitettyyn tietokantaan perustuvissa tallennusratkaisuissa tätä mahdollisuutta ei ole, minkä vuoksi hallintaominaisuudet on toteutettava itse vastaamaan käytettyä tietomallia.

Keskitetyssä tietokannassa hallittavien mallien ja käyttäjien määrä on potentiaalisesti useita kertaluokkia suurempi kuin tiedostopohjaisissa ratkaisuissa. Sekä mallien että käyttäjien suuret määrät luovat tarpeen monipuoliselle tiedon organisoinnille. Käyttäjät muodostavat erilaisia sidosryhmiä, jotka haluavat tarkastella mallinnustietoa erilaisista näkökulmista. Sidosryhmät voivat edustaa esimerkiksi organisaation eri tasoilla toimivia ryhmiä, jotka haluavat tarkastella tietoa eri abstraktiotasoilta. Samaan tapaan mallinnustieto ja sen ryhmittelyperusteet ovat monimuotoisia, kuten mitä kohdetta mallit kuvaavat, minkä tyyppistä ja tasoista mallinnustieto on sekä missä iteraatiossa mallinnustieto luotiin.

Tampereen teknillisessä yliopistossa kehitetyssä mallinnustyökaluja integroivassa Trinity-työkaluympäristössä on käytössä edellä mainittu keskitettyyn tietokantaan perustuva tallennusratkaisu. Ympäristön tietomalli ei ole suoraan yhteensopiva minkään olemassa olevan ratkaisun kanssa, mistä johtuen korkean tason mallinnustiedon hallintaan ja hakuun liittyvät ominaisuudet toteutettiin kokonaisuudessaan itse.

Korkean tason tiedon hallintaa, tiedon hakua sekä muiden yhteisten toimintojen keskittämistä varten toteutettiin hallintakäyttöliittymä, joka integroitiin työkaluksi Trinity-työkaluympäristöön. Se tarjoaa keskitetyn paikan ja yhtenäisen käyttöliittymäkehyksen Trinity-työkaluympäristön yhteisille toiminnoille.

Tässä diplomityössä esitellään hallintakäyttöliittymän tarjoama sovellusalusta sen tarjoamine palveluineen sekä yleiset käyttöliittymäratkaisut, jotka yhdessä luovat laajennettavan työkalun. Työkalun päälle toteutettiin mallinnustiedon hallintaan erikoistunut moduuli, joka tarjoaa mallien ja näkymien hallintaan tarvittavat perustoiminnot. Tiedon hakua varten esitellään kategorisointimekanismi, jota varten toteutettiin siihen erikoistunut moduuli. Moduulitoteutukset esitellään määrittely-, suunnittelu- ja toteutustasolla.

Luvussa 2 esitellään oleelliset korkean tason mallinnustietoon ja työkaluintegrointiin liittyvät käsitteet. Lisäksi luvussa käydään läpi suunnittelussa käytetyt suunnittelu-

mallit ja arkkitehtuurityylit. Luvussa 3 esitellään Trinity-työkaluympäristö, johon hallintakäyttöliittymä integroidaan, ja käydään läpi vaatimuksia korkean tason mallinnustiedon hallinnalle ja toimintojen keskittämiselle. Luvussa 4 esitellään hallintakäyttöliittymän tarjoamat toiminnallisuudet ja käyttöliittymäratkaisut määrittelytasolla. Luku 5 esittelee Trinity-työkaluympäristön aiheuttamat tekniset rajoitteet sekä hallintakäyttöliittymän tekniset suunnitteluperiaatteet ja ratkaisut. Lisäksi luvussa käydään läpi työn kannalta oleellisten moduulitoteutusten yksityiskohtia. Luvussa 6 arvioidaan lopullista ratkaisua yksityiskohtaisesti vaatimusmäärittelyiden perusteella, arvioidaan arkkitehtuurin tarkoituksenmukaisuutta, pohditaan hallintakäyttöliittymän eri roolien merkitystä sekä vertaillaan ratkaisuja kahteen olemassa olevaan järjestelmään. Luku 7 sisältää yhteenvedon.

2. TAUSTA

Tämä luku luo pohjan seuraavissa luvuissa esitetyille asioille. Luvussa käydään lyhyesti läpi oleelliset mallinnuksen ja työkaluintegroinnin peruskäsitteet, sekä esitellään työssä käytetyt arkkitehtuurityylit ja suunnittelumallit.

2.1. Mallinnuskäsitteitä

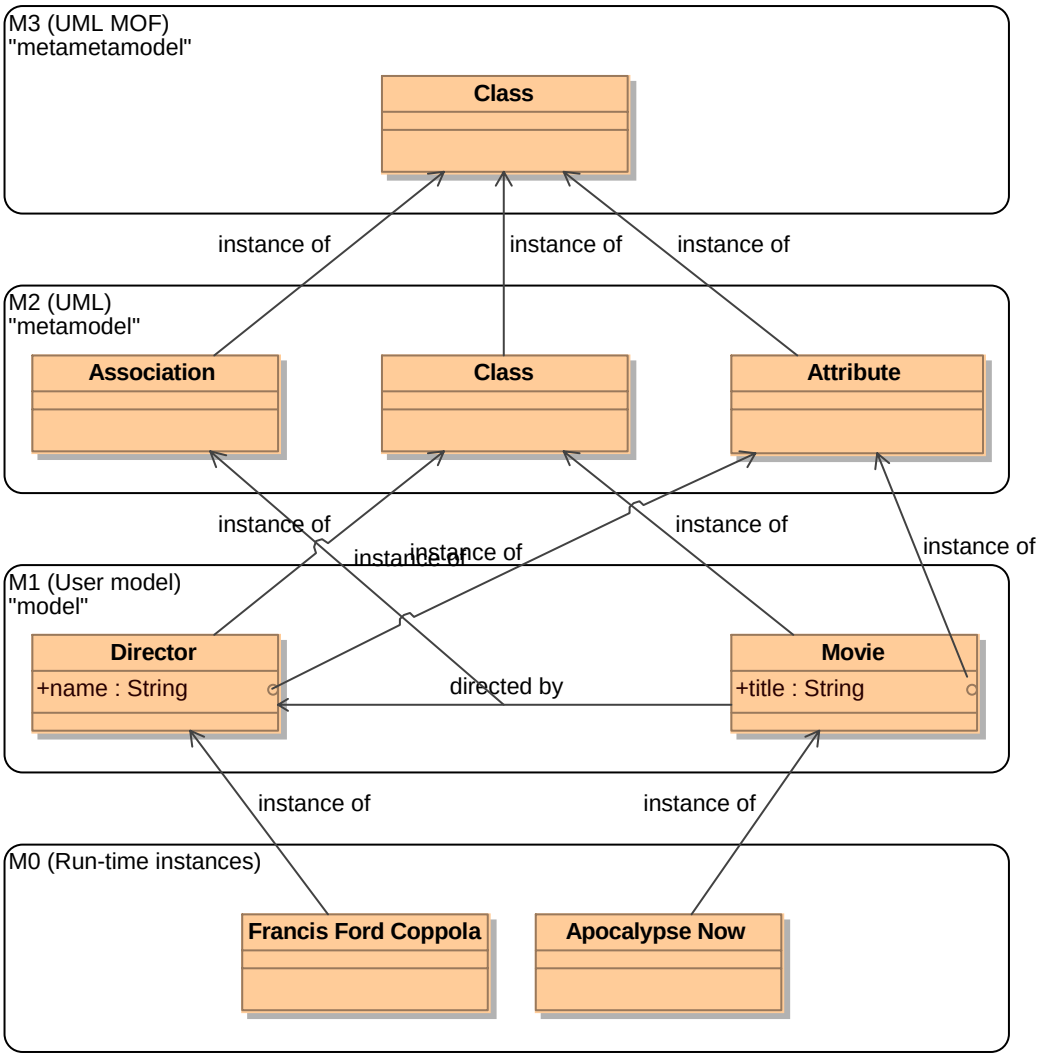
Mallintamisella tarkoitetaan työprosessia, jossa mallinnettavaa kohdetta kuvataan *malliksi*. Stachowiakin mukaan [Sta73, katso Kur07] mallilla on kolme keskeistä ominaisuutta:

- Malli edustaa mallinnettavaa kohdetta, josta ollaan kiinnostuneita.
- Malli on rajoitettu kuvaus kohteen ominaisuuksista. Vain oleelliset asiat mallinnetaan.
- Mallilla on jokin tarkoitus, eli se on pragmaattinen. Tarkoitus voi olla kohteen dokumentointi, analyysi tai suunnittelu.

Tässä työssä mallin määritelmä on UML:n [OMG10] määritelmän kaltainen, jossa malli koostuu *mallielementeistä* ja niiden välisistä *suhteista*. Mallielementit ja suhteet muodostavat varsinaisen loogisen mallinnustiedon.

Malli voi perustua *metamalliin*. Metamalli määrittelee *mallinnuskäsitteet*, joiden avulla mallin sisältöä voidaan kuvata. Myös metamalli voi olla jonkin toisen metamallin mukainen. Tätä seuraavan metatason mallia voitaisiin kutsua metametamalliksi. [Kur07.] Tähän tapaan malli-metamallihierarkia voisi jatkua rajoittamattomasti, mutta käytännön sovelluksissa yleensä rajoitutaan kolmannelle metatasolle, jossa metametamalli määrittelee itsensä (esim. [OMG10]).

Kuva 2.1 esittää esimerkin UML:n MOF:iin perustuvasta nelitasoisesta metamallihierarkiasta. Hierarkian alimmalla tasolla (M0) sijaitsevat ajonaikaiset oliot mallinnettavista kohteista. Ensimmäinen metataso (M1) sisältää käyttäjän tekemää mallinnustietoa malleissa ajonaikaisten olioiden ominaisuuksista. Toinen metataso (M2) sisältää metamalleja, jotka kuvaavat mallinnuskieliä. Mallinnuskielten tarjoamilla käsitteillä käyttäjä voi luoda M1-tason mallinnustietoa. Kolmas metataso (M3) määrittelee itsensä, sekä tavan mallintaa mallinnuskieliä.

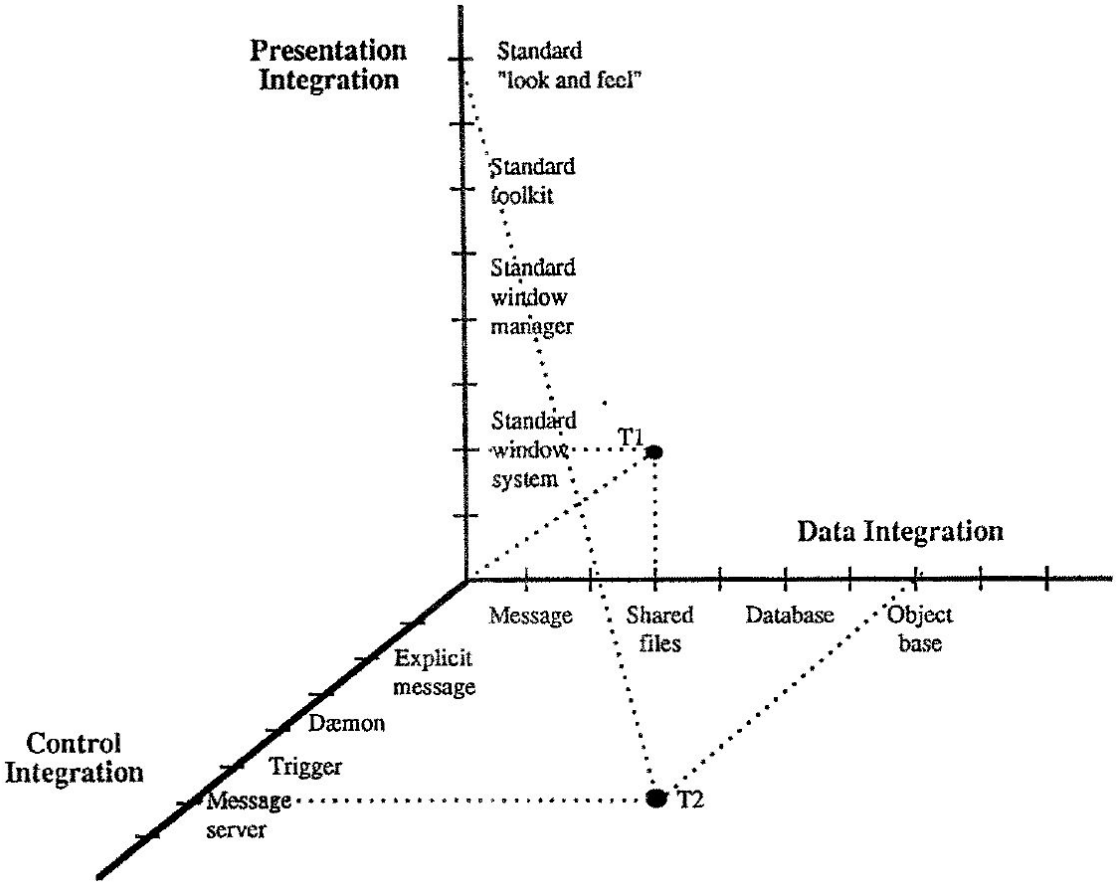


Kuva 2.1: Esimerkki nelitasoisesta metamallihierarkiasta. Mukailtu lähteestä [OMG10].

Näkymä on jonkin mallin osan esitys jostain tietystä näkökulmasta. Näkymiä käytetään mallinnustyössä rajapintana mallintajan ja varsinaisen mallin sisältämän mallinnustiedon välillä mallinnusvaiheessa. Näkymä koostuu *näkymäelementeistä*, jotka edustavat mallielementtejä. Näkymäelementin suhde mallielementtiin on samankaltainen kuin näkymän suhde malliin.

2.2. Työkalujen integrointi

Anthony I. Wasserman esittelee julkaisussaan [Was90] viisi näkökulmaa, jotka tulee ottaa huomioon työkaluintegroinnissa. Kolmen näistä, kontrollin integroinnin, tiedon integroinnin ja esityksen integroinnin, voidaan ajatella muodostavan työkaluintegroinnin kolme ulottuvuutta. Nämä kolme ulottuvuutta voidaan esittää tilaulottuvuuksina (Kuva 2.2), joiden luomaan avaruuteen erilaiset työkalut sijoittuvat. Kaksi muuta näkökulmaa ovat prosessin integrointi ja alustan integrointi.



Kuva 2.2: Työkaluintegroinnin ulottuvuudet [Was90]

Kontrollin integroinnilla tarkoitetaan työkalujen kykyä välittää tapahtumia muille työkaluille, tarvittaessa aktivoiden tapahtuman vastaanottavan työkalun. Esimerkiksi kaavion avaaminen toisessa työkalussa vaatii sekä tapahtumien välitystä työkalujen välillä kaavion osoittamiseksi esimerkiksi tiedostonimen tai yksilöivän tunnisteen avulla, että työkalun aktivointia ennen kaavion avaustoimintoa, mikäli työkalu ei ole valmiiksi aktiivisena.

Tiedon integroinnilla tarkoitetaan integroitavien työkalujen suunnittelemista niin, että ne kykenevät ymmärtämään toisen työkalun tuottamaa tietoa. Esimerkki yksinkertaisesta tiedon integroinnista voisi olla XML-tiedosto, jonka tulkintaan tarvittava tieto, kuten rakenne ja kenttien merkitys, on kaikkien integroitujen työkalujen tiedossa. Näin työkalut kykenevät lukemaan ja tulkitsemaan toistensa luomaa tietoa.

Esityksen integroinnilla tarkoitetaan integroitavien työkalujen käyttöliittymien integrointia. Joissain ympäristöissä, kuten Eclipsessä [EF11a], työkalut integroidaan käyttöliittymän osalta samaan ikkunointijärjestelmän ikkunaan, osaksi suurempaa käyttöliittymää. Toinen vaihtoehto työkalujen esityksen integrointiin on pitää erilaiset työkalut erillisinä, itsenäisinä ja riippumattomina sovelluksinaan, jotka kuitenkin edustavat esimerkiksi jonkun suuremman työnkulun eri vaiheita ja ovat interaktiossa keskenään.

Prosessin integroinnilla tarkoitetaan johonkin päämäärään tähtäävän prosessin eri vaiheiden sitomista toisiinsa integroitavilla työkaluilla. Eri työkalut siis keskittyvät hoitamaan jonkin tietyn vaiheen prosessista ja luovat samalla pohjan seuraavalle vaiheelle.

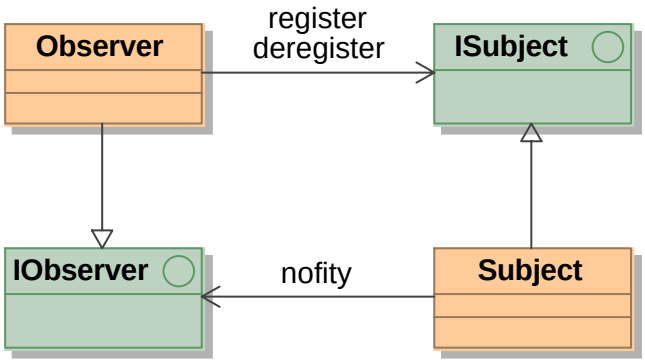
Yhtenäinen alusta, tai ympäristö, on työkalujen integroinnin perusedellytys. Ympäristön on tarjottava työkaluille mahdollisuus päästä käsiksi palveluihin, joiden avulla kontrollin, tiedon ja esityksen integrointi on mahdollista. Alustana voidaan pitää esimerkiksi tyypillistä käyttöjärjestelmää, joka tarjoaa erilaisia palveluita sovellukselle aina tiedostonkäsittelystä käyttöliittymäkehykseen ja prosessienväliseen kommunikointiin.

2.3. Käytetyt suunnittelumallit ja arkkitehtuurityylit

Suunnittelumallit ovat tunnettujen ja käytännössä hyväksi havaittujen ratkaisujen kuvauksia yleisiin ohjelmistojen suunnittelua koskeviin ongelmiin tietyissä tilanteissa. Suunnittelumallissa on kolme olennaista osaa: ongelma, ongelmayhteys ja ratkaisu. Arkkitehtuurityylit ovat pohjimmiltaan suunnittelumallien idean yleistys koko järjestelmän arkkitehtuurin kantavaksi periaatteeksi. [Sha96] [Kos05.] Tässä aliluvussa esitellään työssä esiintyvät suunnittelumallit ja arkkitehtuurityylit.

2.3.1. Tarkkailija-suunnittelumalli

Kuva 2.3 esittää kaavion Tarkkailija-suunnittelumallista ja siihen liittyvien osapuolien keskinäisistä suhteista, sekä rajapinnoista, jotka kätkevät taakseen varsinaiset toteutukset. Tarkkailija-suunnittelumallissa tarkkailija (observer) on kiinnostunut muutoksista jossain kohteessa (subject).



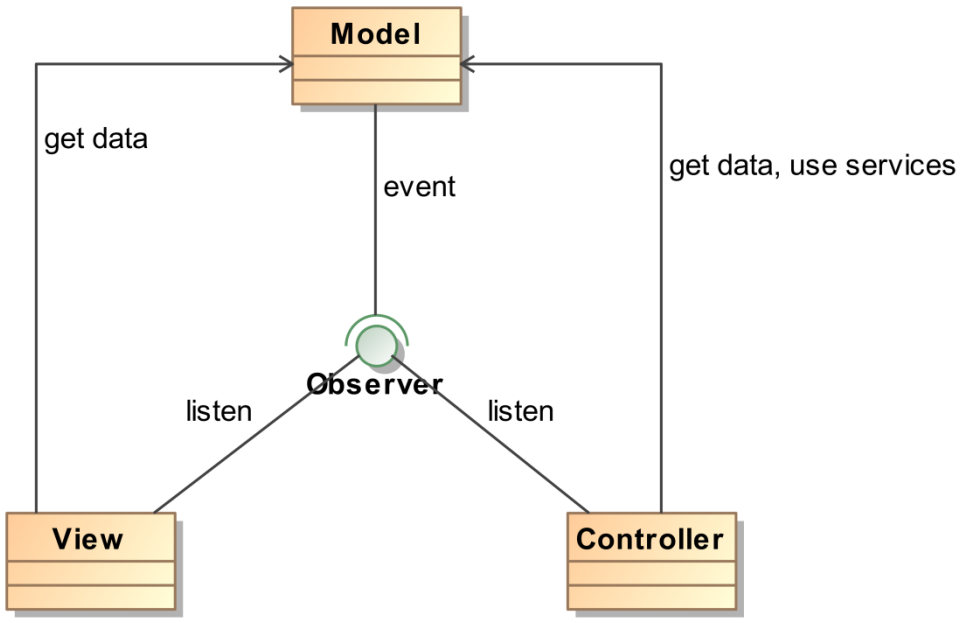
Kuva 2.3: Tarkkailija-suunnittelumalli.

Tarkkailija ilmaisee kiinnostuksensa kohteen muutoksiin rekisteröitymällä kohteeseen tarkkailijaksi. Kohde ylläpitää listaa rekisteröityneistä tarkkailijoista ja ilmoittaa rekisteröityneille tarkkailijoille tapahtuneista muutoksista. Tarkkailija-suunnittelumallissa tarkkailijan ja kohteen liitos on löyhä, eikä kohteen tarvitse olla tietoinen tarkkailijoiden olemuksesta.

2.3.2. Malli-näkymä-ohjain-arkkitehtuuri

[Kos05] kuvaa Malli-näkymä-ohjain-arkkitehtuurin perusajatuksen seuraavasti: ”Malli-näkymä-ohjain-arkkitehtuurin (MVC, model-view-controller) perusajatus on erottaa

käyttöliittymä varsinaisesta sovelluslogiikasta ja -datasta.”. Tämä erotus tekee käyttöliittymän muutoksista helpompia ja helpottaa järjestelmän siirtämistä toiselle graafiselle alustalle. [Kos05.] Kuva 2.4 esittää tässä työssä käytetyn muodon Malli-näkymä-ohjain-arkkitehtuurista.

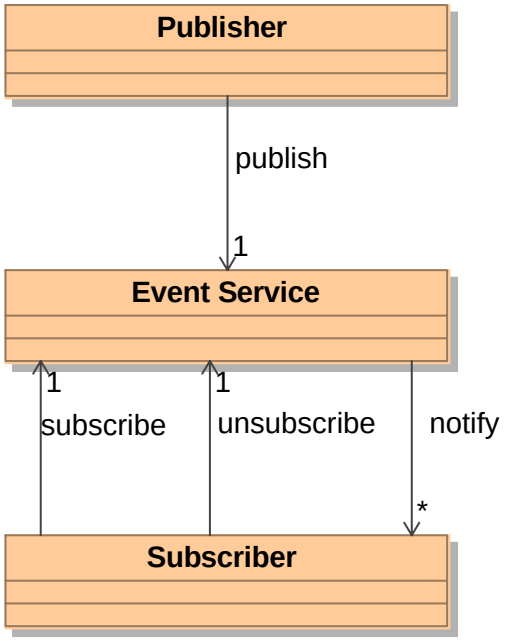


Kuva 2.4: Malli-näkymä-ohjain-arkkitehtuuri.

Arkkitehtuuri koostuu malleista (model), näkymistä (view) ja ohjaimista (controller). Mallit edustavat jotain osaa sovellusdatasta tai sovelluksen loogisesta tilasta. Näkymät edustavat osia sovelluksen käyttöliittymästä. Ohjaimet toimivat mallien ja näkymien välissä pitäen huolta siitä, että ne vastaavat toisiaan. Näkymät ja ohjaimet voivat kuunnella Tarkkailija-suunnittelumallia hyväksikäyttäen mallin muutoksia ja hakea mallilta päivittynyttä tietoa. [Kos05.]

2.3.3. Julkaisija-tilaaja-suunnittelumalli

Julkaisija-tilaaja-suunnittelumalli on viestinvälitykseen liittyvä malli, jossa viestin lähettäjä, tai julkaisija (publisher), ei lähetä viestiä suoraan sen vastaanottajille, tai tilaajalle (subscriber), vaan viesti kulkee eräänlaisen julkaisupalvelun (event service) kautta. Kuva 2.5 esittää kaavion Julkaisija-tilaaja-suunnittelumallista. Julkaisija-tilaaja-suunnittelumallin perusideana on saavuttaa mahdollisimman löyhä liitos viestin lähettäjän ja vastaanottajan välillä. Julkaisija-tilaaja-suunnittelumalli on laajennus Tarkkailija-suunnittelumallista.



Kuva 2.5: Julkaisija-tilaaja-suunnittelumalli

Tilaajilla on mahdollisuus ilmaista kiinnostuksensa viesteihin. Kaikki kiinnostuneet tilaajat saavat ilmoituksen julkaisijan tekemästä viestistä. Julkaisijan ei tarvitse olla tietoinen rekisteröidyistä tilaajista, eikä tilaajien tarvitse olla tietoisia viestien julkaisijoista. [Eug03.]

3. YMPÄRISTÖ JA VAATIMUKSET

Tässä luvussa esitellään Trinity-työkaluympäristö sekä hallintakäyttöliittymän roolit yleisellä tasolla. Lopuksi käydään läpi joukko hallintakäyttöliittymään kohdistuvia vaatimuksia, joiden perusteella varsinaiset ratkaisut suunnitellaan myöhemmissä luvuissa.

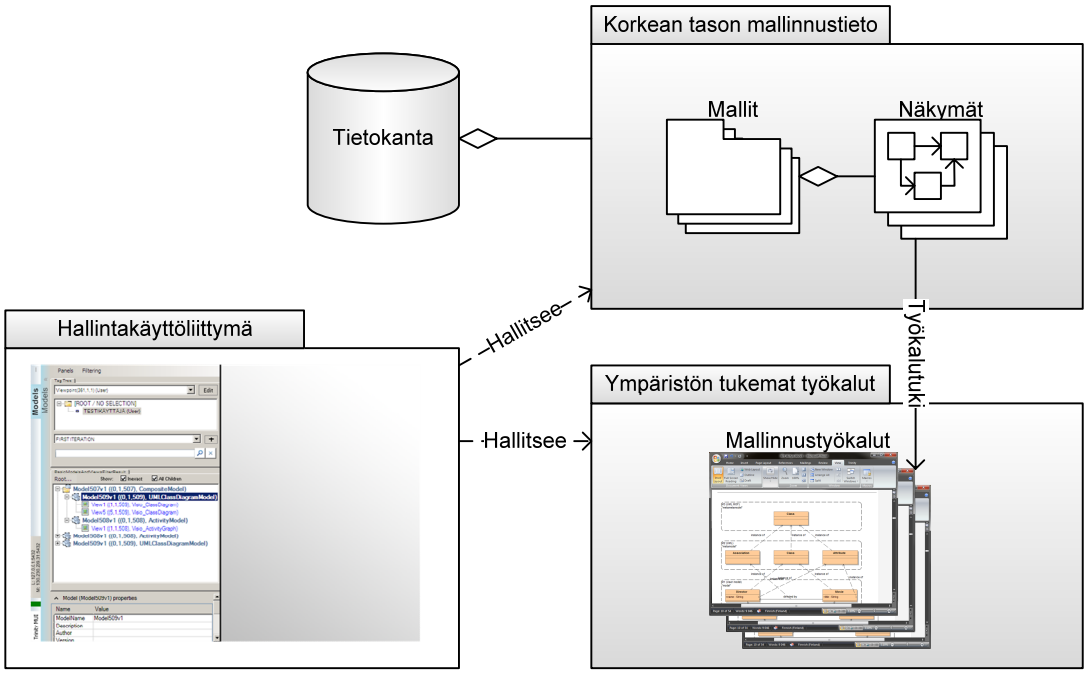
3.1. Trinity-työkaluympäristö ja hallintakäyttöliittymän roolit

Trinity-työkaluympäristö on Tampereen teknillisellä yliopistolla kehitetty monen yhtäaikaisen käyttäjän mallinnustyökaluympäristö. Se tarjoaa käyttäjälle itsenäisen työkalukokonaisuuden, jonka avulla käyttäjä kykenee suoriutumaan mallinnustyön eri vaiheista.

Ympäristö koostuu keskitetystä tietokannasta, kontrolliarkkitehtuurista, mallinnustyökaluista sekä hallintakäyttöliittymästä. Mallinnustyökalut sekä korkean tason hallintatyökalu ovat itsenäisiä, erillisiä sovelluksia, jotka käyttävät tiedon tallentamiseen keskitettyä tietokantaa ja kommunikoivat keskenään kontrolliarkkitehtuurin avulla.

Hallintakäyttöliittymän tarkoituksena on tarjota ympäristössä yksi käyttöliittymä, jonne mallinnustyökalujen ohessa käytettävät yhteiset toiminnallisuudet voidaan keskittää. Keskitetyt toiminnot liittyvät pääasiassa korkean tason tiedon hallintaan, mutta tarpeen mukaan hallintakäyttöliittymään voidaan keskittää muitakin toiminnallisuuksia. Tässä työssä rajaudutaan kuitenkin esittelemään hallintakäyttöliittymän korkean tason mallinnustiedon ja työkalujen hallintaan liittyviä toiminnallisuuksia.

Kuva 3.1 esittää hallintakäyttöliittymän hallitsemat ympäristön osat käyttäjän näkökulmasta. Hallintakäyttöliittymän hallitsema korkean tason mallinnustieto sisältää ympäristön tietokannassa sijaitsevat mallit, mallien sisältämät näkymät ja näiden väliset suhteet sekä näkymiin liittyvän työkalutukitiedon. Hallintakäyttöliittymä hallitsee myös järjestelmään integroitujen mallinnustyökalujen ajonaikaisia instansseja.



Kuva 3.1: Hallintakyttöliittymän hallitsemat kohteet Trinity-työkaluympäristössä

3.2. Mallien ja näkymien hallinta

Hallintakäyttöliittymän korkean tason mallinnustiedon hallinta käsittää Trinity-työkaluympäristön mallien, näkymien ja niiden välisten suhteiden hallinnan. Sekä mallien että näkymien lukumäärän oletetaan voivan kasvaa hyvinkin suureksi, koska kyseessä on monen käyttäjän ympäristö. Ympäristön arvioidaan voivan sisältää kymmeniä tuhansia malleja ja jokaisen mallin arvioidaan voivan sisältää tuhansia näkymiä. Mallien ja näkymien välillä voi olla erilaisia suhteita, joiden päihin liittyy joukko ominaisuuksia. Näitä ominaisuuksia ovat muun muassa rooli ja esitysjärjestys.

Mallien ja näkymien hallintaa varten on olemassa joukko vaatimuksia, joiden perusteella varsinaiset toiminnallisuudet voidaan määritellä. Taulukko 1 listaa ja kuvaa nämä vaatimukset.

Taulukko 1: Mallien ja näkymien hallinnan vaatimukset

Tunniste ja nimi	Kuvaus
V01. Perustietojen esitys	Malleille ja näkymille on määritelty nimi, tyyppi sekä yksilöivä tunniste. Nämä perustiedot tulee olla selkeästi esillä malleja ja näkymiä edustavissa käyttöliittymäelementeissä. **Esimerkiksi** näkymän nimi voisi olla "Korkean tason arkkitehtuurin rakennekuvaus", tyyppi "UML 2.3 Class Diagram" ja yksilöivä tunniste "(123, 4, 5678)".

Tunniste ja nimi	Kuvaus
V02. Perustoiminnot	Malleja ja näkymiä on pystyttävä luomaan, poistamaan ja nimeämään suoraan hallintakäyttöliittymästä. **Esimerkiksi** käyttäjä luo järjestelmään luokkakaavioita varten mallin ja malliin useamman luokkakaavionäkymän. Käyttäjä nimeää luontivaiheessa mallin ja näkymät haluamallaan tavalla. Tämän lisäksi käyttäjälle tulee myöhemmin tarve nimetä mallit ja näkymät uudelleen kuvaamaan paremmin niiden käyttötarkoitusta.
V03. Näkymien ja mallien hierarkkisuus	Näkymät ja mallit muodostavat hierarkkisen rakenteen. Ympäristön sisältämät mallit ja näkymät tulee esittää käyttäjälle niin, että niiden muodostama hierarkkinen rakenne on selvästi esillä. **Esimerkiksi** järjestelmää mallintava malli-näkymä-hierarkia voisi sisältää korkean tason rakenteen kuvaavan mallin, joka koostuu järjestelmän osia kuvaavista alemman tason malleista. Jokainen näistä malleista sisältää erilaisia näkymiä luokkakaavioista interaktionäkymiin, joiden avulla mallinnustyö on tehty.
V04. Suhteiden roolit	Mallien ja näkymien välisten suhteiden päiden määrittelemät roolit tulee ilmaista käyttäjälle selkeällä tavalla. Erilaisia rooleja ovat muun muassa koosteiset roolit sekä mallien riippuvuussuhteita kuvaava kirjastomallirooli. **Esimerkiksi** työkalusovellus voisi riippua järjestelmän yhteisestä tietokantakomponentista. Tätä riippuvuutta voidaan kuvata luomalla kirjastomalliriippuvuutta kuvaava suhde sovellusta kuvaavan mallin ja tietokantakomponenttia kuvaavan mallin välille siten, että tietokantakomponenttia kuvaava malli on kirjastomallin roolissa.
V05. Näkökulmat	Tiedonhaun helpottamiseksi ja luokittelun mahdollistamiseksi mallien ja näkymien joukkoon on pystyttävä luomaan näkökulmia. **Esimerkki:** Näkökulmien avulla käyttäjä voi rajata mallien ja näkymien joukon esityksen kattamaan vain häntä kiinnostavat mallit ja näkymät esimerkiksi organisaatiorakenteen mukaisesti.

Tunniste ja nimi	Kuvaus
V06. Näkökulmien joustavuus	Mallien ja näkymien joukkoon luotavien näkökulmien avulla usean käyttäjän ja organisaation on kyettävä lokeroimaan malleja ja näkymiä joustavasti. **Esimerkki** tyypillisestä käyttötapauksesta on organisaatiorakenteen mukainen lokerointi, jossa organisaatiossa on aliorganisaatioita ja organisaatioihin liittyy useita projekteja, joissa työskentelee useita työntekijöitä. Vaihtoehtoisesti samoja malleja ja näkymiä voitaisiin tarkastella toisesta näkökulmasta, jossa työntekijä rajautuu tarkastelemaan vain itseensä liittyviä malleja jossain projektissa.
V07. Esitysjärjestys	Malleilla ja näkymillä on esitysjärjestys. Tämän esitysjärjestyksen on tultava ilmi myös käyttöliittymästä. Esitysjärjestystä on myös pystyttävä muokkaamaan.
V08. Suhteiden muokkaus	Mallien ja näkymien välisiä suhteita on pystyttävä muokkaamaan suoraan hallintakäyttöliittymästä. Tämä sisältää muun muassa koostesuhteiden muokkauksen, joka puuhierarkiassa merkitsee hierarkiarakenteen muokkausta. **Esimerkiksi** käyttäjä voi luoda suhteen mallien välille raahaamalla mallin hiirellä toisen mallin päälle ja valitsemalla halutun mukaisen roolin.
V09. Mallien ja näkymien raahaus toiseen työkaluun	Malleja ja näkymiä on pystyttävä raahaamaan muihin työkaluihin malli- ja näkymähierarkiasta. **Esimerkiksi** käyttäjä voi raahata olemassa olevan näkymän avoinna olevaan luokkakaavioon. Tässä tilanteessa luokkakaavioon luodaan viite näkymään. Viitteen avulla käyttäjä voi esimerkiksi navigoida viitattuun näkymään suoraan kaaviosta.

3.3. Työkalujen hallinta ja ympäristön tilaindikaattorit

Hallintakäyttöliittymä on vastuussa Trinity-työkaluympäristöön integroitujen työkalujen hallinnasta sekä ympäristön yleisten tilojen esittämisestä. Taulukko 2 listaa ja kuvaa edellä mainittuihin vastuisiin liittyvät vaatimukset.

Taulukko 2: Työkalujen hallinnan vaatimukset

Tunniste ja nimi	Kuvaus
V10. Näkymien avaus	Hallintakäyttöliittymässä esitettyjä näkymiä on pystyttävä avaamaan ympäristöön integroituihin mallinnustyökaluihin.

Tunniste ja nimi	Kuvaus
V11. Työkalun valinta näkymää avattaessa	Samaa näkymätyyppiä voi tukea useampi työkalu. Käyttäjällä on oltava mahdollisuus valita työkalu, jossa näkymä avataan.
V12. Tietokantayhteyden tiedot	Käytössä olevan tietokantayhteyden ominaisuudet tulee esittää selkeästi käyttäjälle. **Esimerkiksi** tietokannan kohdeosoite ja yhteyden tila ovat tällaisia ominaisuuksia.
V13. Avoimet näkymät	Avoinna olevat näkymät tulee ilmaista käyttäjälle selkeästi. Näkymä on avoinna silloin, kun ainakin yksi järjestelmän työkaluista tarjoaa käyttäjälle mahdollisuuden muokata näkymää.
V14. Kirjautuneen käyttäjän tiedot	Järjestelmään kirjautuneen käyttäjän tiedot tulee esittää selkeästi ja yksiselitteisesti. **Esimerkiksi** käyttäjätunnuksen esittäminen näkyvällä paikalla ilmaisee yksiselitteisesti sen, millä käyttäjätilillä järjestelmää ollaan käyttämässä.

3.4. Toimintojen keskittäminen ja käyttöliittymä

Mallinnustyöhön liittyy toistuvia tarpeita, jotka voidaan ratkaista joko toteuttamalla ne erikseen eri työkaluihin tai integroimalla ne yhteen keskitettyyn paikkaan. Hallintakäyttöliittymä on tällainen keskitetty paikka Trinity-työkaluympäristössä.

Kaikkia toimintoja ei kuitenkaan kannata siirtää pois mallinnustyökaluista. Päätös toiminnon keskittämisestä on tehtävä sekä käyttäjän että sovelluskehittäjän näkökulmien muodostamien tarpeiden kompromissina. Käyttäjän näkökulmasta tärkeitä asioita ovat muun muassa käytön helppous ja toiminnon saatavuus erilaisissa tilanteissa. Sovelluskehittäjää kiinnostaa usein toiminnon integrointiin ja toteuttamiseen liittyvän työn määrä.

Taulukko 3 listaa toimintojen keskittämiseen liittyvät vaatimukset hallintakäyttöliittymässä. Nämä vaatimukset eivät ota kantaa yksittäisiin toiminnallisuuksiin ja niiden vaatimuksiin, vaan rajautuu kaikkia toiminnallisuuksia koskeviin hallintakäyttöliittymän yleisiin vaatimuksiin.

Taulukko 3: Toimintojen keskittämisen vaatimukset

Tunniste ja nimi	Kuvaus
V15. Korkea saatavuusaste	Hallintakäyttöliittymän on oltava aina helposti saatavilla, vaikka jokin toinen sovellus olisikin käyttäjän ensisijaisen huomion kohteena. **Esimerkiksi** aina näkyvillä oleva sivupalkki takaa käyttäjän pääsyn hallintakäyttöliittymään muita sovelluksia käytettäessä.
V16. Häiritsemättömyys	Hallintakäyttöliittymä ei saa häiritä käyttäjää tarpeettomasti silloin, kun käyttäjän ensisijaisen huomion kohteena on jokin toinen sovellus.
V17. Johdonmukainen käyttöliittymäkehys	Hallintakäyttöliittymän tulee tarjota yhteisille toiminnoille johdonmukainen käyttöliittymäkehys. Kehyksen tarkoituksena on tarjota käyttäjälle helppokäyttöinen tapa hallita yhteisiä toimintoja hallintakäyttöliittymässä sekä sovelluskehittäjälle helppo tapa integroida uusia toimintoja hallintakäyttöliittymään.
V18. Käyttöliittymäkokonaisuuksien ryhmittely	Hallintakäyttöliittymässä on pystyttävä esittämään yhteisiä toimintoja ryhmiteltyinä käyttöliittymäkokonaisuuksina. **Esimerkki** käyttöliittymäkokonaisuudesta on mallien ja näkymien hallinta sekä siihen liittyvät toiminnot.

3.5. Muut vaatimukset

Seuraavassa taulukossa on listattuna hallintakäyttöliittymän yleisiä, teknispainotteisia vaatimuksia. Nämä vaatimukset tulee ottaa huomioon suunniteltaessa mitä tahansa hallintakäyttöliittymän osaa. Osaa näistä vaatimuksista voidaan myös pitää suosituksena suunnitteluprioriteeteille (laajennettavuus ja dynaamisuus).

Taulukko 4: Muut vaatimukset

Tunniste ja nimi	Kuvaus
V19. Laajennettavuus	Hallintakäyttöliittymän on oltava laajennettavissa helposti uusilla toiminnallisuuksilla.
V20. Dynaamisuus	Hallintakäyttöliittymän on mukauduttava nykyiseen tilanteeseen, johon vaikuttavat muun muassa käyttäjän tekemät valinnat, esitettävän tiedon ominaisuudet sekä työkaluympäristön muiden työkalujen tila.

Tunniste ja nimi	Kuvaus
V21. Mallinnustiedon välitön tallennus tietokantaan	Kaikki ympäristön tietokantaan tallennettava tieto tulee päivittää tietokantaan välittömästi. Erikseen käytettäviä tallennustoimintoja ei tule käyttää, paitsi jos kyseessä on esimerkiksi tiedostojärjestelmään kohdistuva eksplisiittinen tiedon vientitoiminto.
V22. Yhteiset palvelut	Hallintakäyttöliittymän on tarjottava laajennuksille pääsy yhteisiin palveluihin. Näitä palveluita ovat muun muassa Trinity-työkaluympäristön tarjoamat palvelut sekä konfiguraatiotiedon lukeminen.

4. HALLINTAKÄYTTÖLIITTYMÄ

Tässä luvussa esitellään hallintakäyttöliittymän yleisilme sekä korkean tason hallintaan liittyvät ominaisuudet määrittelytasolla. Määrittelyratkaisut perustuvat luvussa 3 esiteltyihin vaatimuksiin.

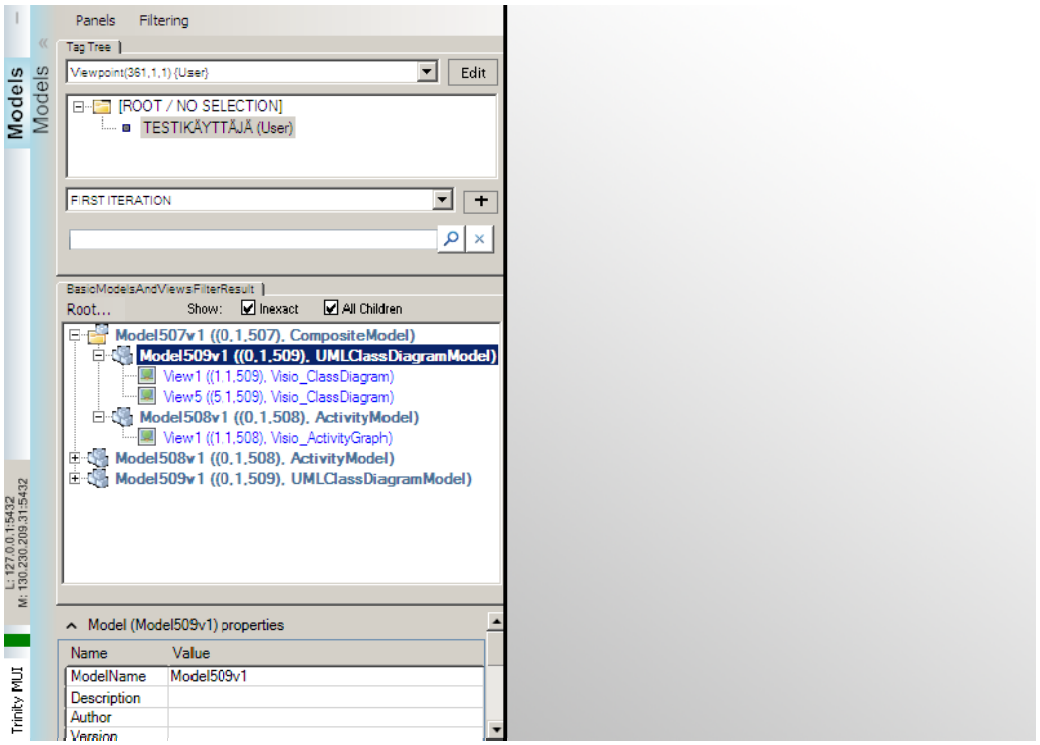
4.1. Yleiset käyttöliittymäratkaisut

Tässä aliluvussa esitellään hallintakäyttöliittymän yleisilme sekä yleiset käyttöliittymäratkaisut. Ratkaisut luovat yhtenäisen pohjan varsinaisille toiminnallisille käyttöliittymäkomponenteille.

4.1.1. Yleiskuva hallintakäyttöliittymästä

Hallintakäyttöliittymä on keskitettyjä toimintoja tarjoava sovellus Trinity-työkaluympäristössä. Se tarjoaa käyttäjälle yhtenäisen paneeleihin ja avautuvaan sivupalkkiin perustuvan käyttöliittymän, jossa keskitetyt toiminnot esitetään.

Erilaiset keskitetyt toiminnot toteutetaan hallintakäyttöliittymään itsenäisinä laajennuksina, joita yhdistelemällä ja koostamalla voidaan luoda erilaisia toimintokokonaisuuksia erilaisiin käyttötarpeisiin. Tyypillisesti yksi toimintokokonaisuus esitetään omana paneelinaan. Kuva 4.1 esittää hallintakäyttöliittymän yleisimmin käytetyn näkymän, jossa on avattuna mallinnustiedon hallintaan liittyvä käyttöliittymäkokonaisuus.

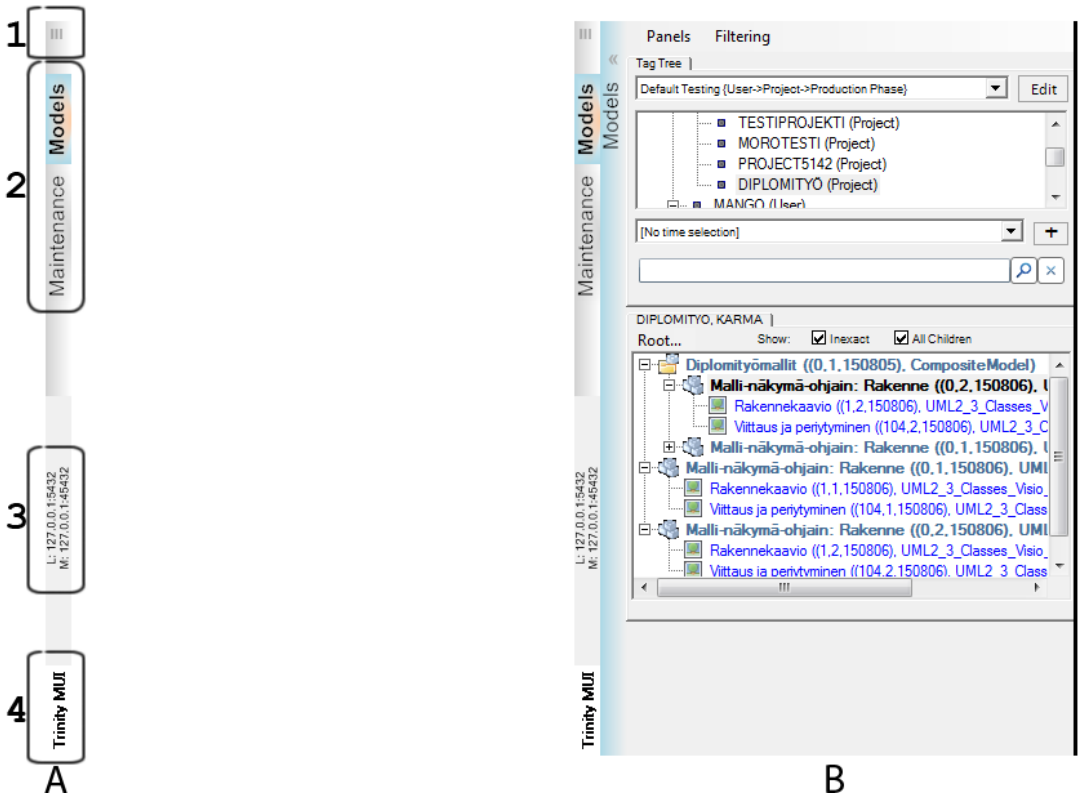


Kuva 4.1: Hallintakäyttöliittymän perusnäkymä

Hallintakäyttöliittymällä on useita rooleja Trinity-työkaluympäristössä aina korkean tason tiedon hallinnasta mallien sisällön esittämiseen ja yleiseen tiedonhakuun. Tässä työssä rajaudutaan käsittelemään hallintakäyttöliittymän yleisiä käyttöliittymä- ja laajennusratkaisuja sekä korkean tason hallintaan liittyvää toimintokokonaisuutta.

4.1.2. Sivupalkki

Hallintakäyttöliittymän lähtökohtana toimii ohut avautuva *sivupalkki*, joka sisältää hallintakäyttöliittymän päävalikon, päänäkymien valintapainikkeet sekä indikaattoreita hallintakäyttöliittymän ja Trinity-työkaluympäristön tilasta. Kuva 4.2 hallintakäyttöliittymän sivupalkin pienennetyssä tilassa (kuvan vasen laita, kohta A) sekä avoimessa tilassa (kuvan oikea laita, kohta B). Pienennetty sivupalkki mahdollistaa pienen käytetyn työpöytäpinta-alan silloin, kun hallintakäyttöliittymä ei ole aktiivisessa roolissa. Vastaavasti avonaisena se pystyy esittämään monimutkaisia ja tilaa vieviä käyttöliittymiä silloin, kun niitä tarvitaan. Sivupalkki vastaa vaatimuksiin korkeasta saatavuudesta (V15) ja häiritsemättömyydestä (V16).

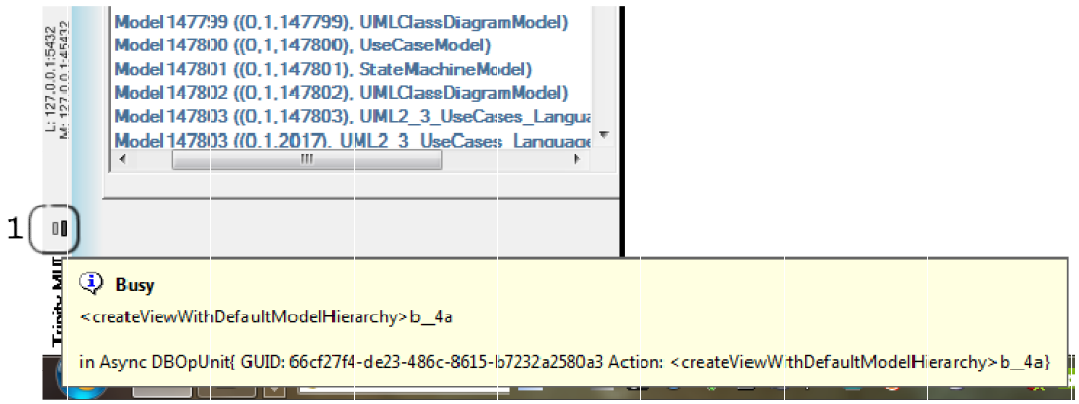


Kuva 4.2: Sivupalkin ominaisuudet

Hallintakäyttöliittymän päävalikko (Kuva 4.2, kohta 4) on yksinkertainen toimintovalikko, jonka avulla voi suorittaa hallintakäyttöliittymän peruskäyttöön liittyviä toimintoja. Näitä ovat muun muassa konfiguraatiotiedoston vaihtaminen ja hallintakäyttöliittymän sammuttaminen.

Päänäkymien valintapainikkeet (Kuva 4.2, kohta 2) kasautuvat päällekkäin sivupalkkiin muodostaen välilehtiä muistuttavan valintatoiminnon. Hallintakäyttöliittymän päänäkymät ovat paneeliryhmiä, joissa paneelit kasautuvat toistensa päälle alkaen sivupalkista. Päänäkymää vaihdettaessa mahdollisesti avoinna ollut toiseen päänäkymään liittyvä paneeliryhmä piilotetaan. Avoinna olevan päänäkymän voi piilottaa myös sivupalkin yläosassa olevalla pienennyspainikkeella (Kuva 4.2, kohta 1). Pienennettynä pienennyspainikkeen tilalle ilmestyy suurennuspainike, jonka avulla valitun päänäkymän voi palauttaa näkyviin. Pienennetyssä tilassa myös päänäkymävalinnan tekeminen valintapainikkeista aiheuttaa käyttöliittymän poistumisen pienennetystä tilasta ja päänäkymän esittämisen. Edellä mainitut ominaisuudet parantavat käyttöliittymän tarjoamien toiminnallisuuksien saatavuutta (V15), tarjoavat piilotusmekanismin (V16) ja mahdollistavat toimintojen jaottelun päänäkymiin, käyttöliittymien ryhmittelyä varten (V18).

Sivupaneelissa esitetään nykyisen konfiguraation mukaiset tietokantaosoitteet tekstimuodossa (Kuva 4.2, kohta 3) (V12) sekä tieto mahdollisesta käynnissä olevasta tietokantaoperaatiosta (Kuva 4.3, kohta 1). Tietokantaosoitteita on nykyisin kaksi; yksi paikalliselle tietokannalle ja yksi ulkoiselle, keskitetylle tietokannalle (kahden tietokantaosoitteen ja niihin liittyvän tietokantatoteutuksen merkitys ei ole tämän työn kannalta tärkeä).

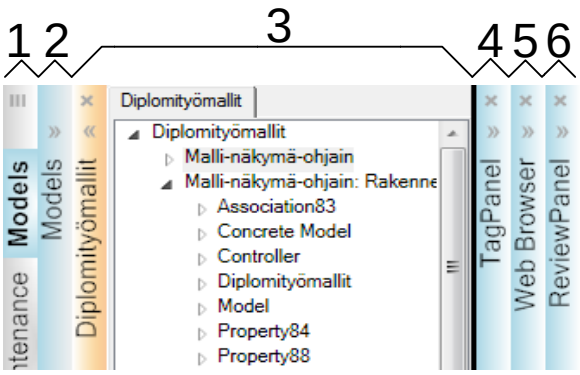


Kuva 4.3: Tietokantaoperaation suoritusindikaattori sivupalkissa

Hallintakäyttöliittymässä tapahtuvan tietokantaoperaation ollessa käynnissä sivupalkissa esitetään työskentelyä kuvaava pieni animaatiokuvake. Tarkempia tietoja operaatiosta esitetään tekstimuodossa kuvakkeen työkaluvihjeenä (engl. "tooltip").

4.1.3. Paneelit

Hallintakäyttöliittymän pääasiallinen esitystapa samaan asiayhteyteen liittyville toiminnoille on koostaa ne paneeliin. Yksi paneeli voisi erimerkiksi edustaa kokonaisuutta, jolla haetaan, järjestetään ja hallitaan järjestelmässä olevia malleja. Paneelirakenteen tarkoituksena on tarjota kaikille hallintakäyttöliittymän komponenteille valmis käyttöliittymäkehys, jonka ansiosta käyttöliittymä pysyy johdonmukaisena. Paneeliratkaisu vastaa suoraan vaatimukseen johdonmukaisesta käyttöliittymäkehyksestä (V17). Kuva 4.4 esittää esimerkkitilanteen hallintakäyttöliittymän paneeleista.



Kuva 4.4: Esimerkkitilanne paneeleista hallintakäyttöliittymässä

Paneeli voidaan pienentää ohueksi palkiksi pois tilaa viemästä (Kuva 4.4, kohdan 3 vasemmassa ylälaidassa näkyvä ”<<”-painike) ja palauttaa avonaiseen tilaansa kun sitä tarvitaan (Kuva 4.4, kohtien 2, 4, 5 ja 6 ylälaidassa näkyvät ”>>”-painikkeet) (V16). Esimerkki avonaisesta paneelista on nähtävissä kuvan kohdassa 3 ja esimerkkejä pienennetyistä paneeleista kohdissa 2, 4, 5 ja 6.

Paneeli voidaan myös sulkea, mikäli sitä ei ole määritelty pysyväksi paneeliksi. Sulkunappulat esitetään paneelien vasemmassa yläreunassa (Kuva 4.4, kohtien 3, 4, 5 ja 6 ylälaidassa näkyvät ”x”-painikkeet).

Avonaisten paneelien kokoa voidaan muuttaa venyttämällä sitä vetämällä hiirellä paneelin mustasta reunuksesta (Kuva 4.4, kohdan 3 oikea reuna). Tämä koko säilyy paneelien muistissa kunnes paneeli suljetaan.

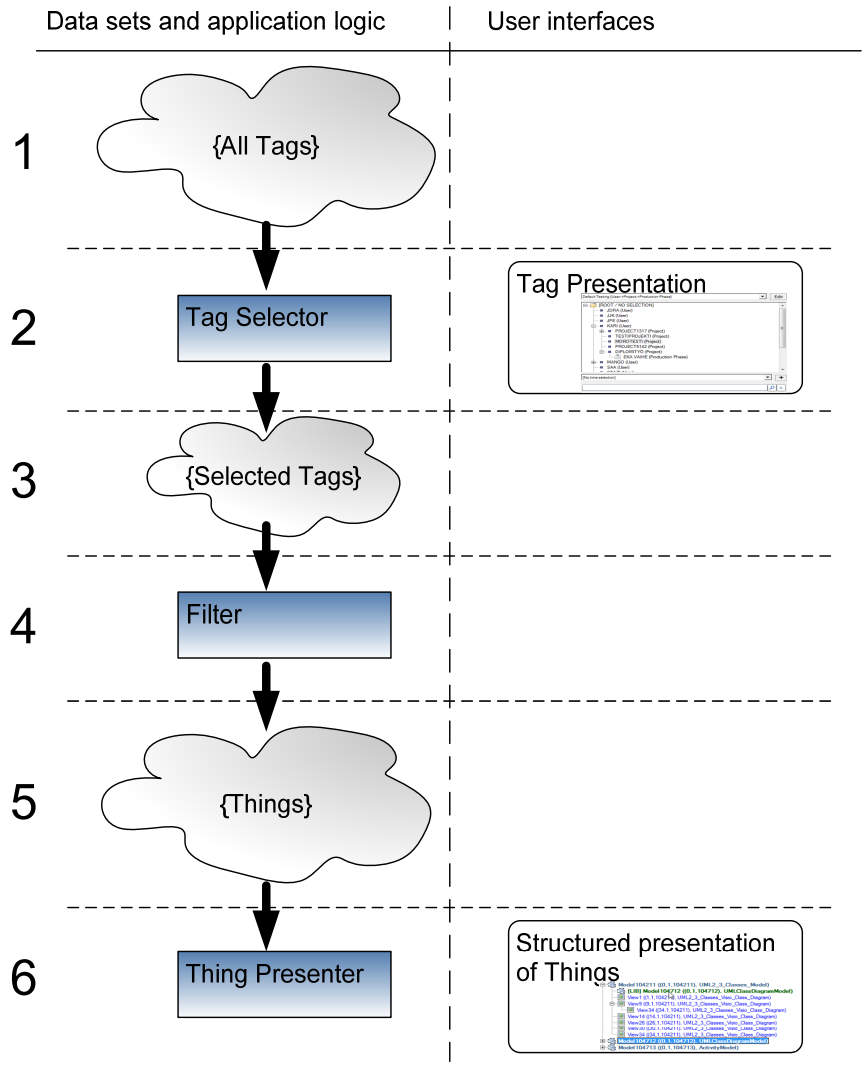
Jos paneeleita on useampia esillä yhtä aikaa, paneelit kasautuvat hallintakäyttöliittymän sivupalkista lähtien peräkkäin vasemmalta oikealle. Paneelit ovat aina toisissaan kiinni. "Paneelikasan" alkuvaiheilla olevan paneelin sulkeminen tai koon muuttuminen aiheuttaa kaikkien seuraavien paneeleiden siirtymisen niin, että paneelit pysyvät kiinni toisissaan.

Hallintakäyttöliittymän sivupalkki (Kuva 4.4, kohta 1) on ulkoasultaan samankaltainen kuin paneelien pienennetyt muodot (Kuva 4.4, kohdat 2, 4, 5 ja 6), mutta toiminnallisesti se poikkeaa paneeleista monella tavalla. Sivupalkki on erikoistunut esittämään indikaattoreiden, päänäkymävalinnan sekä päävalikon lisäksi paneelikasoja, eikä sillä ole varsinaista omaa avattavaa käyttöliittymäkokonaisuutta niiden lisäksi.

4.2. Suodatusprosessi

Hallintakäyttöliittymän yksi tärkeimpiä rooleja on tietokantaan tallennetun tiedon haku. Trinityssä kaikkiin tietokannan entiteetteihin voi liittää *avainsanoja* (engl. ”Tag”), joiden avulla käyttäjä voi hakea ja luokitella tietoa vapaasti.

Kuva 4.5 esittää avainsanoihin perustuvan suodatusprosessin vaiheet kaksijakoisena kaaviona jakautuen ohjelmalogiikkaan ja käyttäjälle esitettäviin käyttöliittymiin. Käyttäjän näkökulmasta prosessi on kaksivaiheinen. Ensin käyttäjä valitsee jostain käyttöliittymästä haluamansa avainsanat, jonka jälkeen jokin toinen käyttöliittymä esittää avainsanojen perusteella tehdyn hakutuloksen.



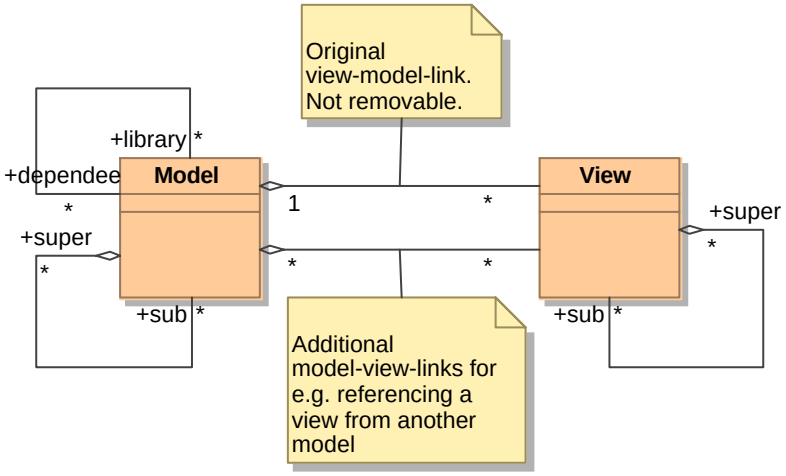
Kuva 4.5: Suodatusprosessin vaiheet

Loogisesti prosessi etenee seuraavasti: Tietokannassa on joukko avainsanoja (All Tags), jotka jokin avainsanojen valintaan erikoistunut komponentti (Tag Selector) esittää käyttäjälle, mahdollisesti rajaten niitä. Kun käyttäjä on tehnyt haluamansa valinnat edellä mainitun komponentin käyttöliittymässä, komponentti toimittaa valintojen perusteella syntyneen joukon avainsanoja suodatinkomponentille (Filter). Suodatinkomponentti hakee saamiensa avainsanojen perusteella joukon tietokantaan tallennettuja entiteettejä (Thing). Yleensä haku tapahtuu yksinkertaisesti valitsemalla kaikki entiteetit, joihin on liitetty jokainen avainsanajoukon avainsanoista. Kun suodatinkomponentti on selvittänyt tulosjoukon ({Things}), se toimittaa joukon komponentille, joka esittää joukon käyttäjälle käyttöliittymässä (Thing Presenter).

4.3. Mallipuu

Yksi hallintakäyttöliittymän rooleista Trinity-työkaluympäristössä on tarjota käyttöliittymäkokonaisuus, jonka avulla käyttäjä voi hallita tietokannassa sijaitsevia malleja ja näkymiä. Tämän käyttöliittymäkokonaisuuden toteuttavaa hallintakäyttöliittymän laajennusta kutsutaan nimellä *mallipuumoduuli*.

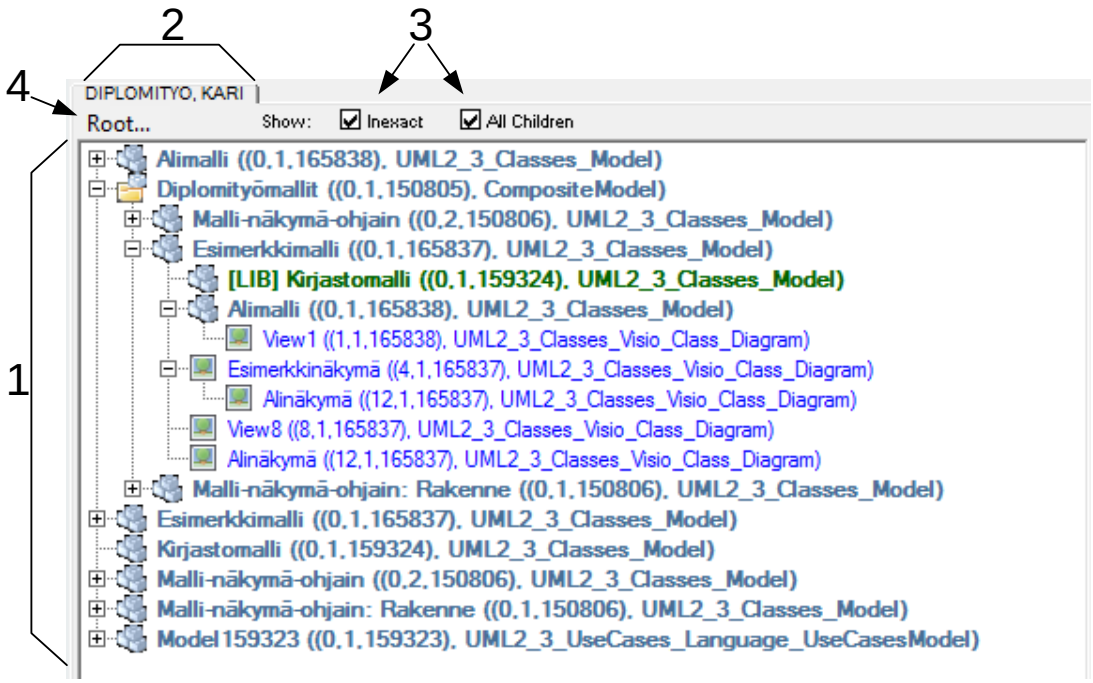
Mallipuun sisällön tuottaminen suoritetaan tietokannassa määriteltyjen mallien ja näkymien välisten koostesuhteiden perusteella. Kuva 4.6 esittää luokkakaaviona mallien ja näkymien välisiä suhteita kuvaavan mallin kielen. Tämä malli on ympäristön tietokannan tuottama ja ylläpitämä malli, joka on rajoitettu kuvaus järjestelmän sisältämistä malleista, näkymistä ja niiden välisistä suhteista.



Kuva 4.6: Kuvaus mallien ja näkymien muodostamasta kielestä

Kirjoitushetkellä mallien ja näkymien malli sisältää koostesuhteet mallista malliin, mallista näkymään sekä näkymästä näkymään. Se sisältää lisäksi riippuvuussuhteita kuvaavat kirjastomallisuhteet mallien välillä. Kooste- ja riippuvuussuhteet ovat käyttäjän vapaasti muokattavissa lukuun ottamatta näkymän alkuperäisen mallin osoittavaa koostesuhdetta. Jokaisella näkymällä on tasan yksi tällainen suhde, eikä sitä voi poistaa.

Kuva 4.7 esittää yleiskuvan mallipuumoduulin käyttöliittymästä. Käyttöliittymä koostuu itse mallipuusta (Kuva 4.7, kohta 1), mallipuun suodatuksessa käytetyistä avainsanoista (Kuva 4.7, kohta 2), mallipuun esitykseen vaikuttavista valinnoista (Kuva 4.7, kohta 3) ja juurivalikosta (Kuva 4.7, kohta 4).



Kuva 4.7: Mallipuukomponentin käyttöliittymä

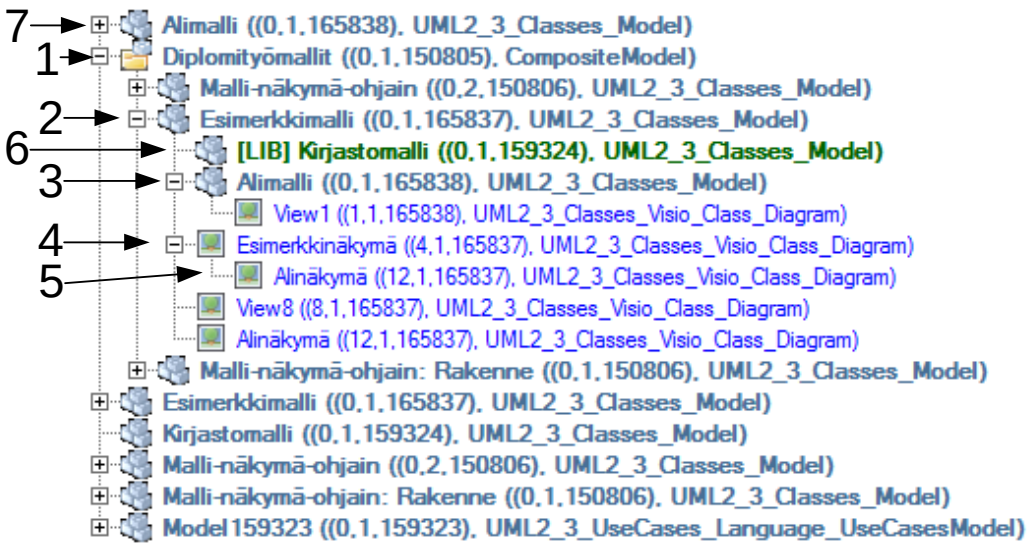
Mallipuu suodatetaan avainsanajoukon perusteella. Avainsanoihin perustuva suodatus antaa käyttäjälle mahdollisuuden luoda mallipuussa esitettävään tietoon näkökulmia (V05). Mallipuun suodatuksessa käytettyjä avainsanoja ei valita kuitenkaan mallipuumoduulissa itsessään, vaan valinta suoritetaan jossain toisessa moduulissa ja valittu avainsanajoukko toimitetaan mallipuumoduulille. Mallipuumoduuli tuottaa esitettävän mallipuun avainsanajoukon perusteella. Mallipuumoduulin voidaan ajatella edustavan avainsanapohjaisen suodatusprosessin suodatinta (Kuva 4.5, kohta 4) ja tulosjoukon esittäjää (Kuva 4.5, kohta 6).

Mallipuun rakenne tuotetaan tietokantaan tallennettujen mallien ja näkymien sekä niiden välisten suhteiden perusteella (V03). Jokainen mallipuun esityksessä esiintyvä puun solmu edustaa jotain tietokannan entiteettiä. Toisin sanoen, mallipuussa esitetään ainoastaan tietokantaan tallennettua tietoa, eikä mallipuussa käytetä esimerkiksi vain esitystä varten luotuja tilapäisiä kansioita. Mallipuuta voi kuitenkin ryhmitellä vapaasti luomalla esimerkiksi yksinomaan koostamiseen tarkoitettuja malleja (Kuva 4.8, kohta 1).

Avainsanavalinnan lisäksi käyttäjä voi vaikuttaa mallipuun esitykseen mallipuun yläpuolella olevilla valintapainikkeilla (Kuva 4.7, kohta 3). Esimerkiksi jälkimmäinen valintapainike (Kuva 4.7, kohta 3, ”Show: All Children”) antaa käyttäjälle mahdollisuuden kytkeä pois alimallien suodatuksen, jolloin mallien suodatusta avainsanojen perusteella sovelletaan ainoastaan puun ylimmälle tasolle. Tämä merkitsee käytännössä sitä, että valinnan ollessa päällä alimallit esitetään aina riippumatta niihin liitetyistä avainsanoista.

Juurivalikko sisältää toiminnot mallien ja näkymien luontiin mallipuun ylimmälle tasolle. Hierarkian ylimmälle tasolle luotavalle näkymälle luodaan automaattisesti sitä tukeva malli.

Kuva 4.8 esittelee mallipuussa esiintyviä erityyppisiä ja erilaisissa rooleissa olevia solmuja. Kaikkia mallipuussa esitettyjä solmuja, paitsi ylimmän tason solmuja, voi järjestää vapaasti hiirellä raahaamalla saman puunhaaran samalla tasolla (V07).



Kuva 4.8: Mallipuu

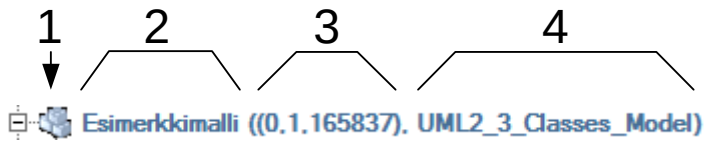
Mallipuun ylimmällä tasolla esitetään kaikki avainsanasuodatuksen läpäisseet mallit. Ylimmän tason mallisolmut (esim. Kuva 4.8, kohdat 1 ja 7) poikkeavat muista mallipuun solmuista siten, että niillä ei ole isäsolmua, johon ne voisivat liittyä. Mallipuulla ajatellaan olevan ainoastaan virtuaalinen juuri, joka vastaa suodatuksessa käytettyjen avainsanojen risteystä. Avainsanojen risteys muodostuu dynaamisesti käyttäjän tekemien valintojen mukaan, eikä sitä vastaa mikään tietokannassa säilytettävä entiteetti. Tämän takia ylimmän tason solmuille ei ole määritelty roolia isäsolmuunsa eikä ennalta määriteltyä järjestystä. Ennalta määritellyn järjestyksen puutteen vuoksi ylimmän tason mallit järjestetään aina aakkosjärjestykseen.

Useimmiten mallipuussa esitettävät mallia tai näkymää edustavat alisolmut ovat koosteisessa alimalli- tai -näkymäroolissa (Kuva 4.8, kohdat 2 ja 3). Koostesuhteeseen liittyvää roolia ei ilmaista erikseen osana solmun nimeä. Erikoisemmat roolit, kuten riippuvuussuhteita kuvaava kirjastomallirooli (Kuva 4.8, kohta 6) ilmaistaan lisäämällä solmun nimen eteen indikaattori roolin tyypistä. Kuva 4.8, kohta 4 osoittaa esimerkin näkymää edustavasta solmusta ja kohta 5 alinäkymää edustavasta solmusta. Roolien esitys vastaa vaatimukseen V04.

Sekä mallit että näkymät saattavat esiintyä mallipuussa useaan kertaan. Yhtä mallia tai näkymää voi siis edustaa useampi mallipuun solmu. Tarkemmin määriteltynä jokainen mallipuun solmu edustaa joko ylimmän tason mallia, eli suodatustulosta, tai jossain roolissa isäsolmunsa edustamaan entiteettiin liitettyä mallia tai näkymää. Esimerkiksi

mallipuussa näkyvä malli nimeltään ”Alimalli” esiintyy mallipuussa kahteen kertaan (Kuva 4.8, kohdat 3 ja 7). Kyseinen malli on alimalli-roolissa kuvan kohdassa 2 esitetylle mallille. Tämän lisäksi se on läpäissyt avainsanasuodatuksen, jolloin se esitetään mallipuun alimmalla tasolla (Kuva 4.8, kohta 7). Esimerkki vastaavasta käyttäytymisestä näkymien osalta on havaittavissa kuvan 4.8 kohdan 5 osoittamalla näkymällä nimeltään ”Alinäkymä”. Näkymä kuuluu ”Esimerkkimalli”-nimiseen malliin, mutta sen lisäksi se on alinäkymä ”Esimerkkinäkymä”-nimiselle näkymälle.

Kuva 4.9 esittää yksittäisen mallipuun solmun esitysmuodon. Esitysmuodosta selviää solmun edustaman entiteetin tärkeimmät perustiedot, kuten nimi, tyyppi ja yksilöivä tunniste (V01). Solmuun liittyy ikoni (Kuva 4.9, kohta 1), josta selviää, onko solmun edustama kokonaisuus malli vai näkymä. Erilaisia malli- ja näkymätyyppejä varten voidaan myös määritellä erilaisia ikoneita parantamaan käytettävyyttä. Ikonin jälkeen esitetään solmun edustaman entiteetin mahdollinen rooli-indikaattori (Kuva 4.8, kohdan 6 ”[LIB]”-osa). Rooli-indikaattorin jälkeen esitetään kokonaisuuden nimi (Kuva 4.9, kohta 2), yksilöivä tunniste (Kuva 4.9, kohta 3) ja joko mallityyppi tai näkymätyyppi, riippuen siitä, onko kokonaisuus malli vai näkymä. Esimerkiksi kuvan 4.9 solmun edustama kokonaisuus on tyypiltään UML2.3:n mukainen luokkamalli (UML2_3_Classes_Model), ja sen nimi on ”Esimerkkimalli”.



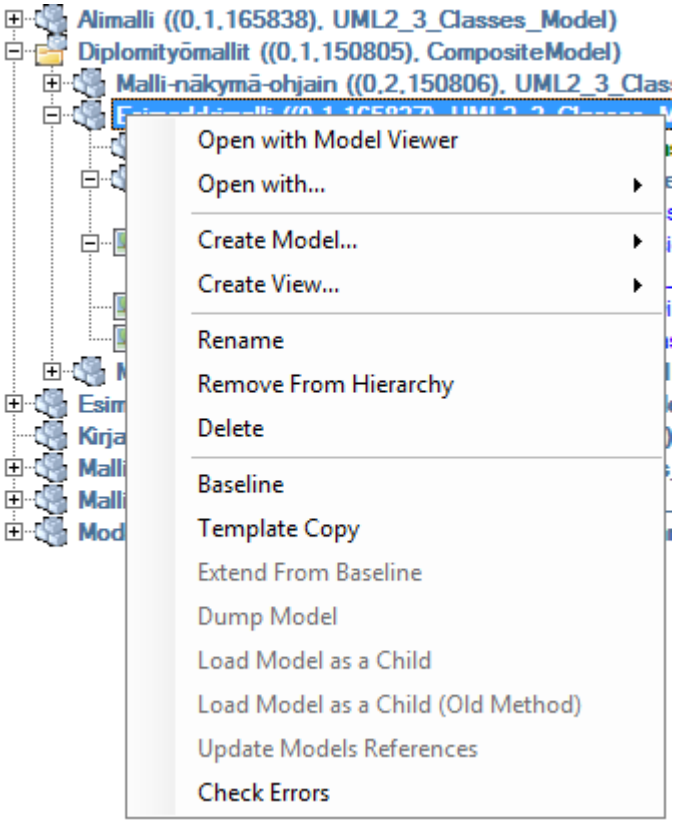
Kuva 4.9: Mallipuun solmun nimen osat

Mallipuun solmuille voidaan suorittaa erilaisia toimintoja solmuista avautuvien pikavalikoiden avulla. Kuva 4.10 esittää pikavalikoiden tarjoamat toiminnot malleille ja Kuva 4.11 näkymille.

Perustoiminnot, kuten uudelleennimeäminen (Rename) ja poisto (Delete), ovat saatavilla sekä malleille että näkymille (V02). Mallit ja näkymät voidaan avata malliselaimeen (Open with Model Viewer), jonka avulla mallin tai näkymään liittyvän mallin sisältöä voidaan tutkia erillisessä paneelissa. Mallit ja näkymät voidaan myös avata järjestelmän tukemassa työkalussa valitsemalla avausvalikosta (Open with...) haluttu työkalu (V10 ja V11).

Erityisesti malleihin liittyviä toimintoja (Kuva 4.10) ovat uuden mallin luonti alimalliksi, näkymän luonti malliin, mallihierarkiasta poisto, baseline-kopiointi, template-kopiointi, sekä muita sekalaisia toimintoja, jotka saattavat riippua mallityypistä. Uuden alimallin luonti suoritetaan mallin luontivalikosta (Create Model...), josta käyttäjä voi valita haluamansa mallityypin. Uusi malli linkitetään automaattisesti nykyisen valitun mallin alimalliksi. Uusi näkymä luodaan vastaavalla tavalla näkymän luontivalikosta (Create View...) ja valitsemalla haluttu näkymätyyppi. Uusi näkymä liitetään valittuna olleeseen malliin. Mallihierarkiasta poisto (Remove From Hierarchy) poistaa linkin alimallin ja koostavan mallin väliltä, jolloin malli poistuu toiminnon nimen mukaisesti

kyseisestä kohdasta mallihierarkiaa. Edellä mainitut luonti- ja poisto-operaatiot, sekä hierarkiasta poistaminen vastaavat vaatimukseen mallien välisten suhteiden muokkaamisesta (V08).



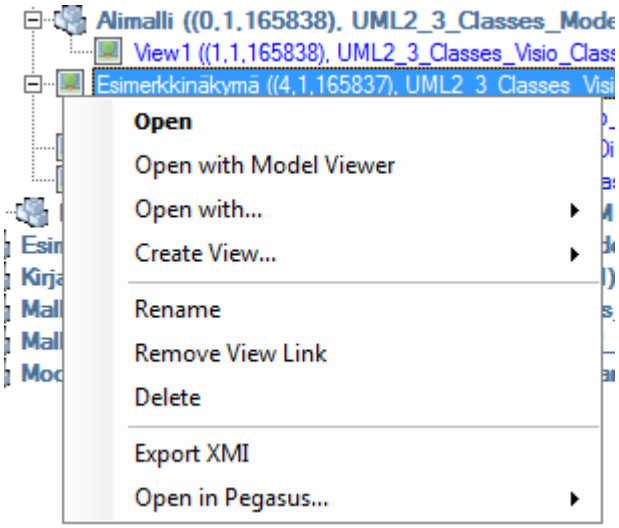
Kuva 4.10: Mallin pikavalikon toiminnot

Baseline-kopiointi (Baseline) kopioi valitun mallin ja koko sen alla olevan hierarkian uusiksi malliversioiksi, jotka viittaavat alkuperäisiin malleihin lähtötilanteenaan. Template-kopiointi (Template Copy) on tavallinen kopiointimekanismi, jolla valittu malli sekä sen alla oleva mallihierarkia yksinkertaisesti kopioidaan uudeksi hierarkiaksi. Edellä mainittujen toimintojen lisäksi mallin pikavalikko sisältää joitain muitakin sekalaisia toimintoja, mutta ne eivät ole tämän työn kannalta tärkeitä, eikä niitä esitellä yksityiskohtaisesti.

Malleille voidaan suorittaa kohdennettuja toimintoja hiiren kakkosnapilla raahaamalla. Tällaisia toimintoja ovat muun muassa olemassa olevan mallin linkitys toiseen malliin erilaisiin rooleihin, kuten alimalliksi tai kirjastomalliksi. Lisäksi baseline- sekä template- kopiointitoiminnot voidaan suorittaa kohdennetusti, jolloin toimintojen tuloksena syntyneet uudet mallit linkitetään automaattisesti kohteen alle. Edellä mainitut operaatiot vastaavat vaatimukseen V08.

Kuva 4.11 esittää näkymään liittyvät pikavalikon toiminnot. Näkymä voidaan avata näkymän tyyppiin liitetyssä oletustyökalussa (Open) (V10). Avaus oletustyökalussa voidaan suorittaa myös kaksoisnapsauttamalla näkymää edustavaa solmua. Näkymäsolmuun voidaan mallisolmun tapaan luoda halutun tyyppinen näkymä (Create

View...), joka tässä tilanteessa kuitenkin liitetään lisäksi valitun näkymän alinäkymäksi. Alinäkymältä voi poistaa linkin koostavaan näkymään näkymälinkin poistotoiminnolla (Remove View Link). Näkymä voi olla linkitettynä myös muihin malleihin kuin siihen missä se varsinaisesti sijaitsee, jolloin ne esitetään mallipuussa kuin ne olisivat myös kyseisten mallien sisältämiä näkymiä. Nämä ”ylimääräiset” linkit malleihin voidaan poistaa samalla näkymälinkin poistotoiminnolla kuin alinäkymälinkitkin (Remove View Link). Edellä mainitut luonti ja poisto-operaatiot, sekä hierarkiasta poistaminen vastaavat vaatimukseen mallien ja näkymien välisten suhteiden muokkaamisesta (V08). Pikavalikon alaosassa on esitettynä myös muita sekalaisia toimintoja, jotka eivät kuitenkaan ole tämän työn kannalta tärkeitä, eikä niitä esitellä yksityiskohtaisesti.



Kuva 4.11: Näkymän pikavalikon toiminnot

Olemassa olevia näkymiä voidaan liittää muiden näkymien alinäkymiksi hiiren kakkosnäppäimellä raahaamalla. Alinäkymät voivat olla myös eri mallissa kuin niitä koostavat näkymät. Lisäksi näkymiä voidaan liittää muihin malleihin kuin missä ne varsinaisesti sijaitsevat. Tällaisten ”ylimääräisten” koostesuhteiden avulla eri malleissa sijaitsevia näkymiä voidaan koostaa samaan paikkaan tiedon löytämisen helpottamiseksi tai muiden tiedon organisointitarpeiden tyydyttämiseksi. Edellä mainitut operaatiot vastaavat vaatimukseen V08.

4.4. Kategorisointi

Yksinkertaisiin avainsanoihin perustuva tiedon suodatus soveltuu hyvin tilanteisiin, joissa avainsanojen lukumäärä on pieni. Trinity-työkaluympäristössä avainsanoja käytetään pääasiallisena hakumekanismina kaikelle tiedolle, minkä vuoksi avainsanojen lukumäärä voi kasvaa suureksi. Suuren avainsanajoukon käsittely muuttuu helposti omaksi käytettävyysongelmakseen. Tietoon halutaan pystyä luomaan myös joustavia näkökulmia avainsanojen avulla. Tässä aliluvussa esitellään ratkaisu vaatimuksiin mallien ja näkymien hallinnan näkökulmista (V05) ja näiden näkökulmien joustavuudesta (V06).

4.4.1. Kategoriat ja kategoriatyypit

Yksinkertaiset avainsanat eivät ole tarpeeksi monipuolisia sellaisenaan vastaamaan Trinity-työkaluympäristön tiedon organisointitarpeisiin. Avainsanoilta puuttuu erityisesti niiden käyttötarkoitusta kuvaava tyyppi. Tätä varten määritellään laajennettu avainsana nimeltään *kategoria,* jolla on tyyppi, ja joka mahdollistaa tyyppitietoon perustuvat *näkökulmat.*

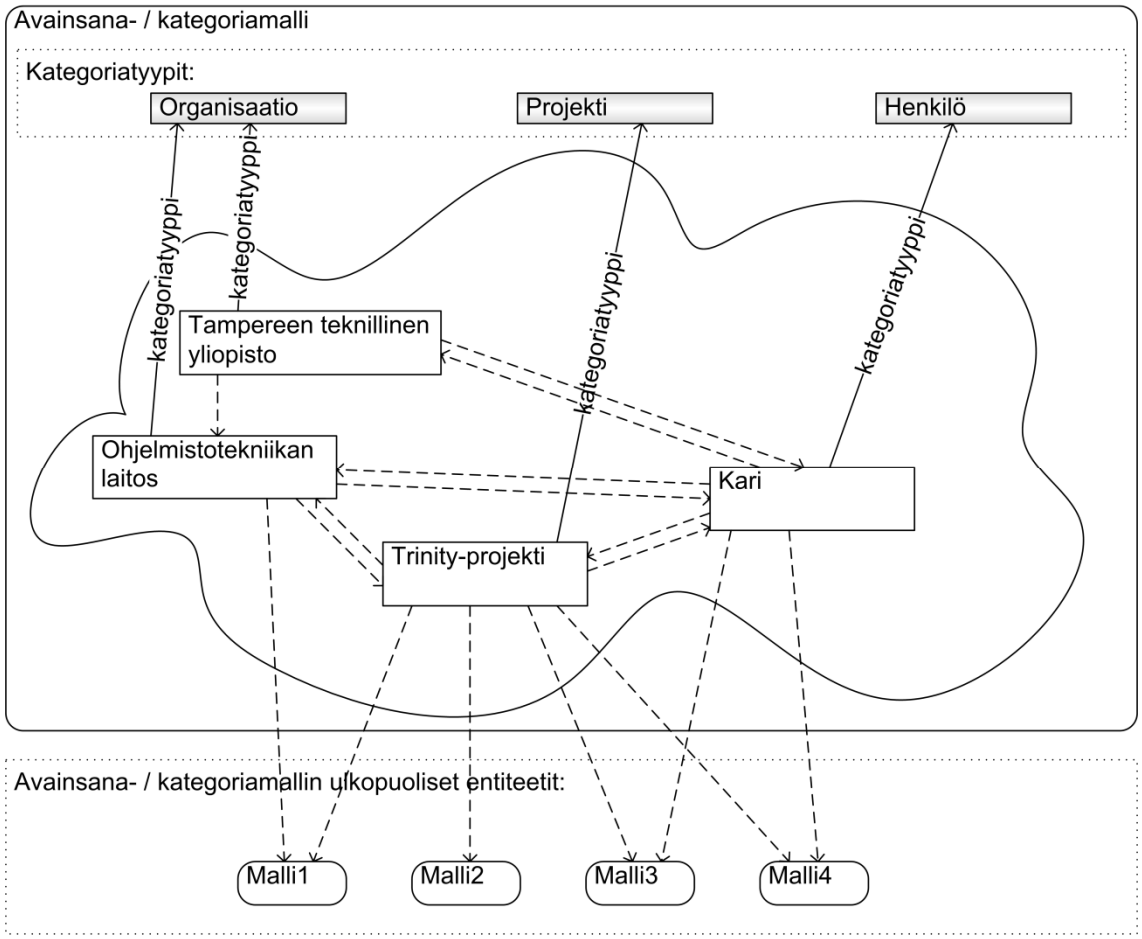
Kategorian voidaan ajatella olevan eri asioita yhdistävä suhde, joka määrittelee jonkin yhtenevyyden niiden välillä. Esimerkiksi tuotteet "omena" ja "päärynä" voitaisiin liittää kategoriaan "hedelmä", jolloin saadaan muodostettua niitä yhdistävä lokerointiperuste.

Kategorian tyypistä käytetään nimityksiä *kategoriatyyppi* tai *avainsanaan liitetty tyyppi.* Myös hallintakäyttöliittymän käyttöliittymissä termit kategoria ja avainsana ovat tässä työssä usein samanarvoisia. Kategoria-käsitettä käytetään erityisesti silloin, kun esitysmuoto on jollain tavalla rakenteinen ja kategoriatyypillä on merkitys.

Kategoriatyyppi ilmaisee kategorian käyttötarkoituksen ja liittää sen johonkin ymmärrettävään, usein tosimaailmalliseen, käsitteeseen. Esimerkkejä kategoriatyypeistä ovat ”Organisaatio”, ”Projekti”, ”Tuote”, ”Käyttäjä”, ”Aika”, ”Mallin nimi” ja ”Mallin tyyppi”. Tyypillinen kategoria ja siihen liittyvä kategoriatyyppi voisi olla esimerkiksi kategoria nimeltään ”TTY”, johon on liitetty kategoriatyyppi nimeltään ”Organisaatio”. Käyttäjän on kategoriatyypin perusteella helpompi ymmärtää kategorian käyttötarkoitus. Kategoriatyyppi avaa myös uusia mahdollisuuksia sovelluksille käsitellä kategorioita. Sovellus voi esimerkiksi esittää tietyn tyyppiset kategoriat käyttäjälle intuitiivisemmalla tavalla. Esimerkiksi ”Aika”-tyyppisten kategorioiden käsittely voi poiketa huomattavasti ”Organisaatio”-tyyppisten kategorioiden käsittelystä ja esitystavasta. Aika mielletään usein lineaariseksi ulottuvuudeksi, kun taas organisaation voidaan ajatella olevan hierarkkinen rakenne.

Trinity-työkaluympäristössä kategorisointitieto säilötään sen kuvaamiseen erikoistuneeseen malliin. Malli sisältää kategoriat, kategoriatyypit sekä kategorioihin liittyvät suhteet. Suhteita on kahdenlaisia: kategorian kategoriatyypin osoittava suhde sekä kategoriaan liitetyn tietokantaentiteetin osoittava suhde. Yhteen kategoriaan liittyy tasan yksi kategoriatyyppi ja nolla tai useampia tietokantaentiteettejä, mukaan lukien muut kategoriat. Nämä suhteet ovat nykyisessä määrittelyssä yksisuuntaisia.

Kuva 4.12 esittää esimerkin kategoriamallissa sijaitsevasta kategoriaverkosta. Kuvassa esitetyn pilven yläpuolella on joukko kategoriatyyppejä, joiden tyyppisiä kategorioita kategoriapilvessä voi olla. Pilven sisällä on joukko kategorioita, joista jokainen viittaa yhteen kategoriatyyppiin ja useampaan tietokantaentiteettiin. Näiden viittausten (suhteiden) perusteella voidaan päätellä esimerkiksi, että projektiin "Trinity-projekti" liittyy henkilö "Kari", johon puolestaan liittyy kaksi mallia ("Malli3" ja "Malli4").



Kuva 4.12: Esimerkki kategoriaverkosta

4.4.2. Näkökulmat kategoriaverkkoon

Koska kategorioiden välillä voi olla mielivaltainen määrä suhteita, kategoriat muodostavat kokonaisuutena monimutkaisen verkkorakenteen, jonka esittäminen käyttöliittymässä on haastavaa. Eräs ratkaisu kategoriaverkon helpompaan käsittelyyn on rajoittaa rakennetta ja näytettävää sisältöä näkökulmilla, jotka määrittelevät näytettävän puurakenteen, johon kategorioiden verkko projisoidaan. Puurakenteen esitys käyttöliittymissä on helppoa ja antaa samalla käyttäjälle helposti ymmärrettävän kuvan kategorioista ja niiden välisistä suhteista käyttäjän itsensä määrittelemässä näkökulmassa.

Kategoriaverkon projisointiin puurakenteeksi tarkoitetun näkökulman määritelmä tässä diplomityössä on seuraava: Näkökulma on järjestetty lista kategoriatyyppejä, jotka määrittelevät puun tuottamisessa läpi käytävät kategoriatyypit, aloittaen juuritasolta.

Kategoriaverkon projisointi näkökulman avulla puurakenteeksi suoritetaan seuraavasti: Jokaisen kategoriapuussa olevan kategorian K alle valitaan kategoriat, joihin K viittaa, ja jotka ovat joko samaa kategoriatyyppiä kuin K tai näkökulman määrittelemän seuraavan kategoriatyypin tyyppisiä kategorioita. Jos K toistuu omana lapsisolmunaan (esimerkiksi kategorioiden muodostaman silmukkarakenteen johdosta), uutta K:n ilmentymää ei lisätä puuhun.

Esimerkki näkökulmasta voisi olla [Organisaatio -> Projekti -> Henkilö]. Kategoriapuun projisointi kuvan 4.12 esittämästä kategoriaverkosta edellä mainitulla näkökulmalla tuottaa listauksen 1 kuvaaman puurakenteen kategorioita:

Listaus 1: Kategoriapuu

```
* Ohjelmistotekniikan laitos
  * Trinity-projekti
    * Kari
* Tampereen teknillinen yliopisto
  * Ohjelmistotekniikan laitos
    * Trinity-projekti
      * Kari
```

Näkökulma määrittelee puun ensimmäiselle tasolle kategoriatyypin "Organisaatio". "Ohjelmistotekniikan laitos" sekä "Tampereen teknillinen yliopisto" vastaavat tätä kategoriatyyppiä, jolloin ne tulevat valituksi puun ensimmäiselle tasolle. "Tampereen teknillinen yliopisto"-kategoriasta on suhde toiseen "Organisaatio"-tyyppiseen kategoriaan "Ohjelmistotekniikan laitos", joka lisätään alisolmuksi. Seuraava näkökulman määrittelemä kategoriatyyppi on "Projekti". Ainoastaan "Ohjelmistotekniikan laitos"-kategoriasta kulkee suhde "Projekti"-tyyppisen kategoriaan, "Trinity-projektiin", joka lisätään kaikkien "Ohjelmistotekniikan laitos"-kategorian ilmentymien alisolmuksi. Viimeinen näkökulman määrittelemä kategoriatyyppi on "Henkilö". "Trinity-projekti"-kategoriasta voidaan navigoida "Henkilö"-tyyppiseen kategoriaan "Kari", joka lisätään alisolmuksi kaikille "Trinity-projekti"-kategorian ilmentymille alisolmuksi.

Näkökulmien määrittäminen Trinity-työkaluympäristössä on käyttäjän vastuulla. Vapaasti määriteltävien näkökulmien avulla käyttäjät kykenevät hakemaan ja tarkastelemaan ympäristön kategorioita erittäin joustavalla tavalla.

4.4.3. Kategorioiden valinta kategoriapuusta

Kirjoitushetkellä kategoriapuussa on yksi tapa suorittaa kategoriavalinta. Listaus 2 kuvaa pseudokoodina yksinkertaistetusti kategoriapuun solmujen valinta-algoritmin.

Listaus 2: Kategoriajoukon valinta kategoriapuun solmuista

```
selectedCategories = empty set of Categories;
currentNode = user selected Node in CategoryTree;

while( currentNode != null )
{
   selectedCategories.add( currentNode.Category );
   currentNode = currentNode.Parent;
}
```

Kategoriapuun solmun valitseminen tulkitaan siten, että valitun solmun lisäksi valituksi tulee jokainen isäsolmu, joka tulee vastaan kun puun haaraa seurataan juureen saakka. Kategoriavalintajoukkoon lisätään jokainen kategoriapuun valittua solmua vas-

taava kategoria. Muodostunut kategoriavalintajoukko toimitetaan suodatusprosessin (Kuva 4.5) seuraavaan vaiheeseen, eli avainsanapohjaiselle suodattimelle (Filter).

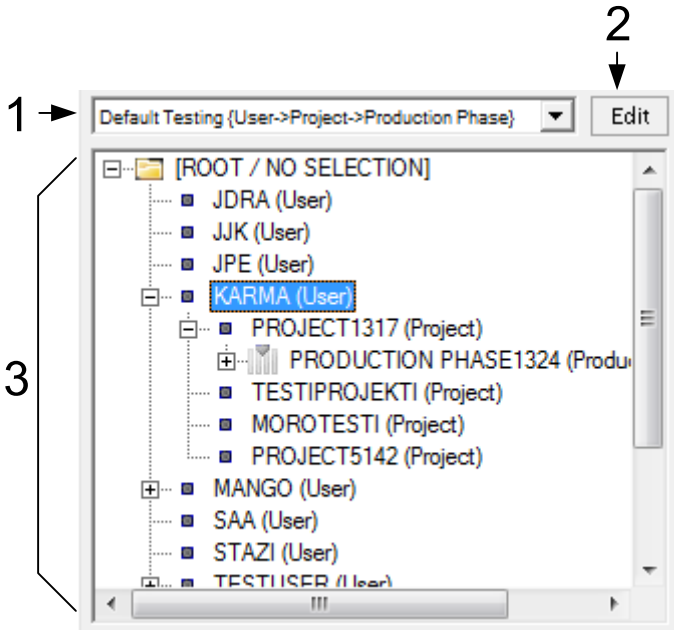
Esimerkki: Valitsemalla listauksen 1 esittämästä kategoriapuusta kolmannella rivillä esiintyvän solmun, joka edustaa "Kari"-kategoriaa, tuottaisi kategoriavalintajoukon {"Kari", "Trinity-projekti", "Ohjelmistotekniikan laitos"}. "Tampereen teknillinen yliopisto"-kategoria ei sisälly valintajoukkoon, koska se ei tule vastaan polulla valitusta solmusta puun juureen. Tämän jälkeen valintajoukko toimitetaan avainsanapohjaiselle suodattimelle, joka tässä työssä tarkoittaa kohdassa 4.3 esiteltyä mallipuuta. Mallipuussa esitellyn suodatuslogiikan perusteella tuloksena saataisiin tyhjä joukko tietokantaentiteettejä, sillä yksikään kuvan 4.5 entiteeteistä ei ole liitetty kaikkiin tämän kategoriavalintajoukon kategorioista. Mikäli valintajoukko olisi {"Kari", "Trinity-projekti"}, tulokseksi saataisiin joukko {"Malli3" ja "Malli4"}.

4.4.4. Kategorisointikäyttöliittymä

Hallintakäyttöliittymässä kategorioiden tarkastelua ja hallintaa varten on olemassa kuvan 4.14 mukainen käyttöliittymä. Käyttöliittymä koostuu näkökulman valintaosasta (Kuva 4.13, kohta 1), näkökulmien muokkauspainikkeesta (Kuva 4.13, kohta 2) sekä kategoriapuusta (Kuva 4.13, kohta 3).

Näkökulman valinta tehdään alasvetovalikolla, joka listaa käytettävissä olevat näkökulmat. Alasvetovalikon valintavaihtoehto nimetään sen edustaman näkökulman mukaan siten, että alussa esitetään näkökulman nimi ja sen jälkeen aaltosulkeissa näkökulman sisältämä kategoriatyyppiketju.

Kategoriapuu esittää näkökulman perusteella tuotetun puurakenteen kategorioista kohdan 4.4.2 mukaisesti. Käyttäjä voi luoda ja poistaa kategorioita avaamalla puun solmusta hiirellä pikavalikon ja valitsemalla haluamansa toiminnon. Luomisvaihtoehtoja on kaksi. Käyttäjä voi luoda joko näkökulman kategoriatyyppiketjun mukaisen seuraavan kategoriatyypin tyyppisen kategorian tai saman tyyppisen kategorian kuin valittu kategoria. Uusi kategoria liitetään valittuun kategoriaan luonnin yhteydessä ja se ilmestyy valitun solmun alisolmuksi.



Kuva 4.13: Kategoriapuu ja näkökulmat

Kategoriapuun solmuissa esitetään solmun edustaman kategorian nimi ja tyyppi. Toisin kuin kohdassa 4.3 esitellyssä mallipuussa, kategoriapuussa ei ole tarvetta esittää puun solmuja vastaavien entiteettien yksilöivää tunnistetta, sillä kategoriat yksilöidään nimen ja tyypin kombinaationa.

Kategorioita, ja avainsanoja yleisesti, voidaan lisäksi hallita laajemmin erillisen avainsanapaneelin avulla. Avainsanapaneelin avulla voidaan luoda, sekä liittää ja irrottaa avainsanoja tietokantaentiteeteistä. Avainsanapaneelissa tehdyt muutokset päivittyvät välittömästi myös edellä mainittuun kategoriapuuhun.

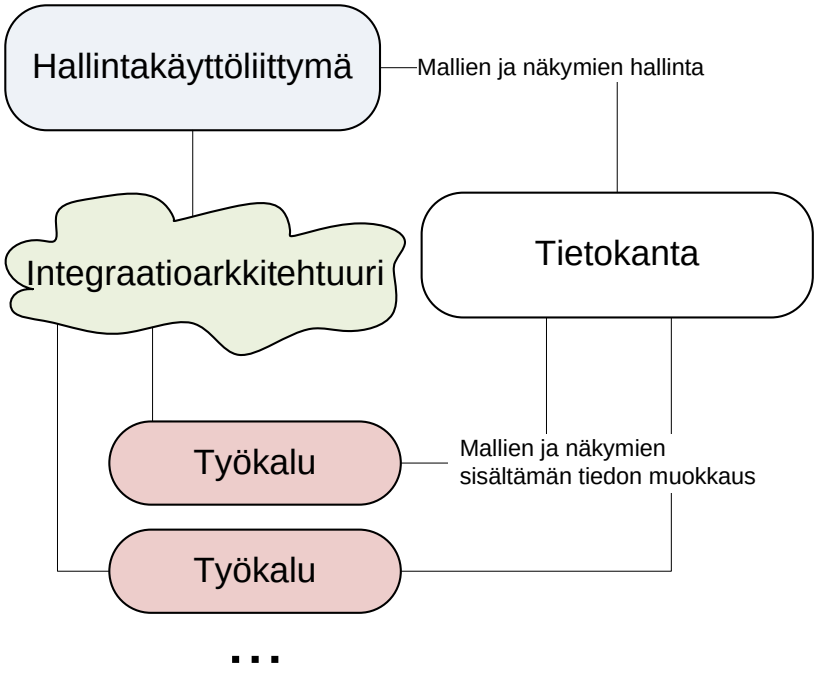
Myös näkökulmia voidaan muokata erillisen käyttöliittymän kautta, joka tarjoaa käyttäjälle mahdollisuuden luoda uusia näkökulmia, nimetä ja poistaa olemassa olevia sekä määritellä näkökulman sisältämän kategoriatyyppiketjun. Ketjussa olevien kategoriatyyppien järjestystä voi myös muuttaa. Muutokset näkökulman määrittelyssä kuvastuvat välittömästi kategoriapuussa, mikäli kyseinen näkökulma on valittu kategoriapuun perustaksi.

5. SUUNNITTELU JA TOTEUTUS

Tässä luvussa esitellään Trinity-työkaluympäristön yleinen arkkitehtuuri ja sen tarjoamat palvelut. Ympäristön esittelyn jälkeen käydään läpi hallintakäyttöliittymän suunnittelussa vaikuttaneet suunnitteluperiaatteet ja toteutustekniikat, sekä suunnittelun tuloksena syntyneet ratkaisut arkkitehtuuritasolla. Lisäksi luvussa esitellään tämän diplomityön kannalta oleellisten moduulitoteutuksien yksityiskohtia.

5.1. Trinity-työkaluympäristön yleinen arkkitehtuuri

Trinity-työkaluympäristön yleinen arkkitehtuuri voidaan jakaa karkeasti neljään kokonaisuuteen kuvan 5.1 mukaisesti. Nämä kokonaisuudet ovat yhteinen mallipohjainen tietokanta, kommunikaatio- ja ohjauskanavana toimiva integraatioarkkitehtuuri, mallinnustyökalut ja hallintakäyttöliittymä. Kokonaisuuksien merkitys ja toiminta on kuvattu omissa alikohdissaan tarkemmin.



Kuva 5.1: Trinity-työkaluympäristön korkean tason rakenne

5.1.1. Tietokanta

Trinity-työkaluympäristössä on yleisperiaate, jonka mukaan kaikki mallinnustieto tulee tallentaa ja lukea yhteisestä tietokannasta. Tämä tietokanta perustuu relaatiotietokantaan ja tarjoaa mallinnustiedon tallentamiseen mallipohjaisen tietorakenteen olio-relaatiorajapintana toimivan tietokantakomponentin avulla. Tietokantakomponentti kät-

kee täysin taakseen tietokannan relaatiomallin, minkä vuoksi sovelluksen ohjelmoijan ei tarvitse olla tietoinen tietokannan toteutustavasta.

Tietokantakomponentin tarjoama oliomalli sisältää kaksi tietokannassa määriteltyä metatasoa. Korkein metataso, metametataso, sisältää metametamallin (ks. liite 1), joka on ainoa pysyvä malli Trinity-työkaluympäristön tietokannan tietomallissa. Mallinnuskielet määritellään metametamallin tarjoamien käsitteiden avulla. Mallinnuskielet puolestaan määrittelevät mallinnuskäsitteet eri tyyppisille malleille, joita käyttäjät voivat luoda hallintakäyttöliittymässä. Käyttäjän luomat mallit sisältävät käyttäjän luomaa mallinnustietoa, jota käsitellään kielten määrittelemien näkymien avulla. Näkymä voi olla esimerkiksi UML2.3:n [OMG10] luokkakaavionäkymä, jota muokataan jollain sitä tukevalla työkalulla.

5.1.2. Integraatioarkkitehtuuri

Trinity-työkaluympäristö tarjoaa työkalujen välille integraatioarkkitehtuurin, joka on agenttiarkkitehtuuri-kommunikointimallin mukainen. Agenttiarkkitehtuurin toiminta perustuu agentteihin [Muh11], jotka liikkuvat paikkojen ja alueiden välillä ja suorittavat niissä toimintoja [Muh11]. Agenteilla on myös tila. Agentit pystyvät suorittamaan monimutkaisiakin toimintaketjuja päämääriensä saavuttamiseksi ja kantamaan mukanaan informaatiota. Kaikki Trinity-työkaluympäristöön integroidut työkalut kykenevät lähettämään ja vastaanottamaan sekä luomaan agentteja agenttiarkkitehtuurin tarjoamien rajapintojen avulla. Agentit ovat lisäksi ohjelmointikieliriippumattomia, jonka johdosta työkalujenkaan ei tarvitse olla toteutettu samalla ohjelmointikielellä.

Paikka (Location) on agenttiarkkitehtuurissa käytetty termi, joka edustaa paikkaa, joka tarjoaa palveluita, joita agentit voivat käyttää. Paikat edustavat tyypillisesti käyttöliittymiä tai rajapintoja tietokantoihin tai muihin sovelluksiin. Alue (Area) on ryhmä paikkoja, jotka yleensä sijaitsevat yhdellä tietokoneella. Agentit liikkuvat paikkojen välillä tavoitehakuisesti suorittaen ennalta määriteltyjä tehtäviä. [Muh11.]

Trinity-työkaluympäristön kontrolliarkkitehtuuri on kirjoitushetkellä rajoittunut yhdelle työasemalle ja koostuu yhdestä alueesta ja useammasta paikasta. Jokaista työkalua sekä hallintakäyttöliittymää varten on olemassa yksi paikka. Näiden lisäksi esimerkiksi tietokannan hallinnalle sekä tiedostosynkronoinnille tiedostojärjestelmän ja tietokannan välillä on omat paikkansa.

5.1.3. Työkalut

Trinity-työkaluympäristöön integroidut työkalut toimivat rajapintana käyttäjän ja tietokannan välillä. Ne ovat pääasiassa mallinnustyökaluja ja tarjoavat käyttäjälle tavan muokata mallinnustietoa näkymien avulla. Nämä työkalut voivat olla ominaisuuksiltaan ja olemukseltaan hyvin monimuotoisia ja erilaisia. Työkaluja yhdistää kuitenkin Trinity-työkaluympäristön tarjoamat palvelut; yhteinen tietomalli ja integraatioarkkitehtuuri, joiden ansiosta työkalut kykenevät toimimaan yhdessä.

Työkalut erikoistuvat tyypillisesti tulkitsemaan, esittämään ja muokkaamaan tietokannassa määriteltyjen kielien mukaisia näkymiä ja mallien sisältöä. Kielet eivät ole kuitenkaan sidottuja työkaluihin, vaan eri työkalut voivat tukea samoihin kieliin perus-

tuvaa mallinnusta. Työkalut voivat myös käsitellä samaa mallia ja näkymää yhtä aikaa, ja saavat reaaliajassa tiedon tietokannassa tapahtuvista muutoksista tietokantakomponentin kautta.

Eräs esimerkki Trinity-työkaluympäristön integroidusta työkalutoteutuksesta on Microsoft Visioon perustuva UML2.3 –mallinnuskieltä tukeva mallinnustyökalu [Pel10]. Tässä työssä esitelty hallintakäyttöliittymä on erikoistapaus Trinity-työkaluympäristöön integroidusta työkalusta.

5.2. Trinity-työkaluympäristön työkaluille asettamat rajoitteet

Kohdassa 5.1.1 esitetyn periaatteen mukaan kaikki mallinnustieto tulee tallentaa ympäristön tarjoamaan tietokantaan. Mallinnustietoa ei tule tallentaa tai ladata esimerkiksi tiedostosta, ellei kyseessä oli erityinen tiedon vienti- tai tuontitoiminto.

Mallinnustyökalujen on kyettävä ymmärtämään ja käsittelemään tietokannan tarjoamaa metamallipohjaista tietomallia. Kaikki mallinnustieto tulee tallentaa Trinity-työkaluympäristön tietokannan määrittelemän tietomallin mukaisesti.

Mikäli työkalu haluaa kommunikoida tai muuten hallita muita ympäristön työkaluja, sen tulee käyttää tähän ympäristön tarjoamaa kontrolliarkkitehtuuria. Tämä kontrolliarkkitehtuuri esiteltiin kohdassa 5.1.2.

Trinity-työkaluympäristöä voi käyttää yhtä aikaa monta käyttäjää monelta työasemalta. Ympäristöön integroitavien työkalujen on siis kyettävä päivittämään tilaansa ympäristössä tapahtuvien muutosten perusteella. Työkalun on yleisesti kyettävä toimimaan monen käyttäjän ympäristössä.

5.3. Hallintakäyttöliittymän suunnitteluperiaatteet ja yleinen arkkitehtuuri

Tässä aliluvussa esitellään hallintakäyttöliittymän teknisessä suunnittelussa tehdyt ratkaisut, käytetyt toteutustekniikat sekä yleinen rakenne. Lopuksi esitellään hallintakäyttöliittymän moduulien suunnitteluperiaatteet.

5.3.1. Toteutustekniikat

Hallintakäyttöliittymän suunnitteluvaiheessa Trinity-työkaluympäristön tarjoamat palvelut tukivat ainoastaan Microsoft .NET -sovelluskehystä [MS11a]. Tästä syystä hallintakäyttöliittymän sovelluskehykseksi valittiin edellä mainittu sovelluskehys.

.NET tukee pääasiassa kahta ohjelmointikieltä; C#- [MS11b] ja Visual Basic -ohjelmointikieliä. Näistä vaihtoehdoista toteutuskieleksi valittiin C#, jonka vahvuuksia Visual Basic -kieleen verrattuna ovat muun muassa tyyppiturvallisuus ja tehokkuus sekä tuttuus Trinity-projektin jäsenille.

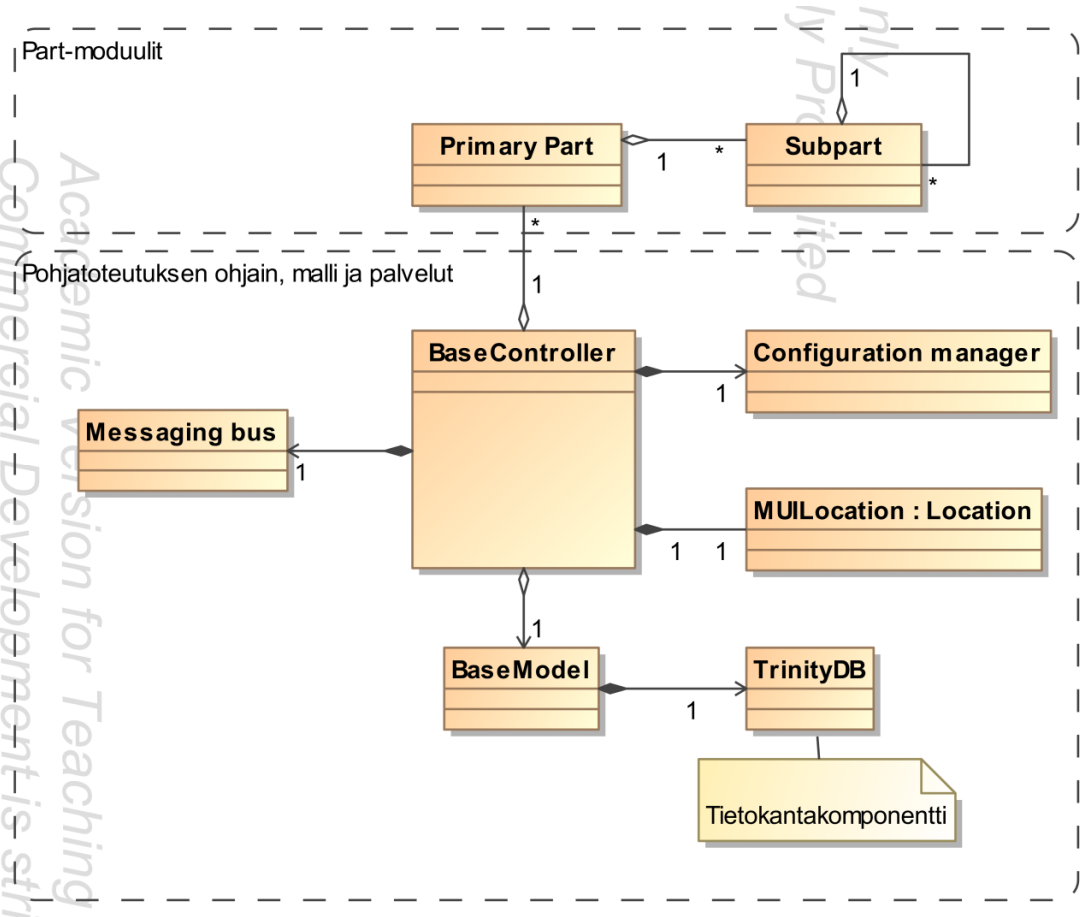
Käyttöliittymäkehykseksi valittiin .NET:n tarjoama Microsoft Windows Forms –käyttöliittymäkehys [MS10a]. Suunnitteluvaiheessa modernimpi Windows Presentation Foundation (WPF) -kehys [MS10b] oli myös saatavilla ja mahdollinen valinta, mutta sen todettiin olevan liian keskeneräinen. .NET tarjoaa kuitenkin Windows Formsin ja

WPF:n yhteensovittamiseksi sovitinmekanismin [MS11c], jonka avulla Windows Forms -käytöliittymät voivat sisältää WPF-elementtejä ja toisinpäin.

5.3.2. Rakenne ja laajennusmekanismi

Hallintakäyttöliittymä koostuu sovelluskehyksenä toimivasta pohjatoteutuksesta ja sen päälle kiinnittyvistä moduuleista (V19). Pohjatoteutus tarjoaa moduuleille pääsyn Trinity-työkaluympäristön palveluihin (V22), yhteisen Julkaisija-tilaaja-suunnittelumallin mukaisen tiedotuskanavan moduulien väliseen tapahtumienvälitykseen sekä keskitetyn tavan tallentaa paikallisesti tietoa pysyväismuistiin. Moduulit erikoistuvat hoitamaan jotain tiettyä toiminnallisuuskokonaisuutta ja tarjoavat käyttäjälle tarkoitukseen sopivan käyttöliittymän. Esimerkiksi aikaisemmin esitellystä mallipuusta ja sen toiminnoista voisi luoda yhden moduulin, joka tarjoaa käyttäjälle puumaisen käyttöliittymän ja toimintalogiikan mallipuun sisällön hallintaan.

Hallintakäyttöliittymän rakenne mukailee MVC-arkkitehtuurityyliä. Luokkakaavio arkkitehtuurista on esitetty kuvassa 5.2. MVC-arkkitehtuuri on luonnollinen ja usein käytetty arkkitehtuurityyli etenkin käyttöliittymäsovelluksissa, ja toimii siten hyvänä lähtökohtana hallintakäyttöliittymän suunnittelulle.



Kuva 5.2: Hallintakäyttöliittymän rakennemalli

Hallintakäyttöliittymän pohjatoteutuksen ohjain (BaseController) tarjoaa moduuleille asetustenhallinnan (Configuration) sekä hallintakäyttöliittymälle tarkoitetun ilmentymän (MUILocation) agenttiarkkitehtuurin Paikka-käsitteen toteuttavasta luokasta. Ohjain tarjoaa myös rajapintansa kautta pääsyn käytössä olevaan yleiseen malliin (BaseModel).

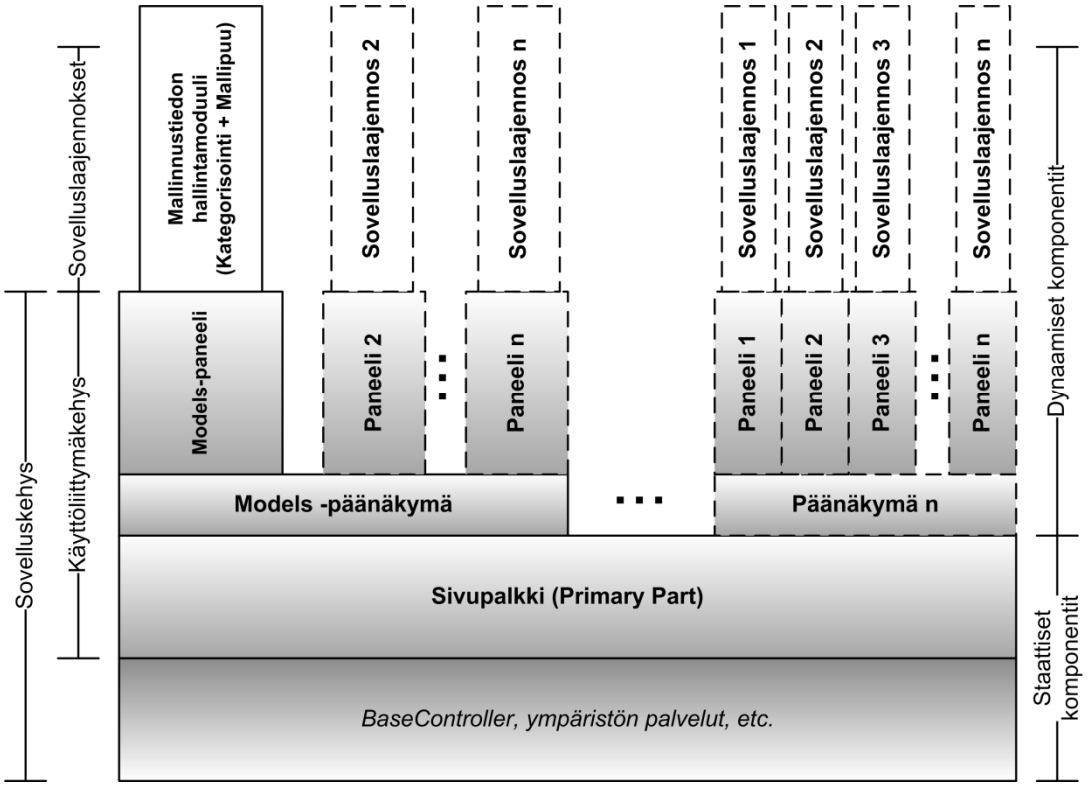
Asetustenhallinta on keskitetty paikallisten asetusten säilyttämiseen tarkoitettu komponentti. Sen avulla moduulit voivat tallentaa käyttäjän tekemiä valintoja ja muuta istuntokohtaista tietoa käyttöliittymissä. Asetustenhallinta sarjallistaa ja tallentaa pysyväismuistiin siihen tallennetut asetukset istunnon päätyttyä. Asetustenhallinta lataa sarjallistetut asetukset muistiin automaattisesti uuden istunnon alettua, jolloin ne ovat taas moduulien käytettävissä.

Hallintakäyttöliittymän Paikka-ilmentymä (MUILocation) tarjoaa hallintakäyttöliittymän osille helpon tavan kutsua agenttiarkkitehtuurin kautta suoritettavia yleisesti käytettyjä toimintoja. Sen avulla voidaan agenttien avulla esimerkiksi avata näkymiä ja käynnistää muita työkaluja.

Hallintakäyttöliittymän pohjatoteutuksen ohjain sisältää lisäksi Julkaisija-tilaaja-suunnittelumallin mukaisen viestiväylän. Viestiväylän avulla eri moduulit kykenevät lähettämään ja vastaanottamaan tietoa hallintakäyttöliittymässä tapahtuvista tapahtumista, jotka voivat potentiaalisesti koskea mitä tahansa muuta moduulia. Esimerkiksi käyttäjän tekemä valinta missä tahansa osassa hallintakäyttöliittymää voi olla kiinnostava tapahtuma moduuleille, joiden tarkoitus on esittää käyttäjälle valittuna olevan elementin tietoja.

Mallin roolia rakenteessa edustaa hallintakäyttöliittymän yleinen malli (BaseModel), jonka rajapinnan kautta on mahdollista päästä käsiksi tietokantakomponenttiilmentymään (TrinityDB), joka on yhteinen kaikille hallintakäyttöliittymän osille.

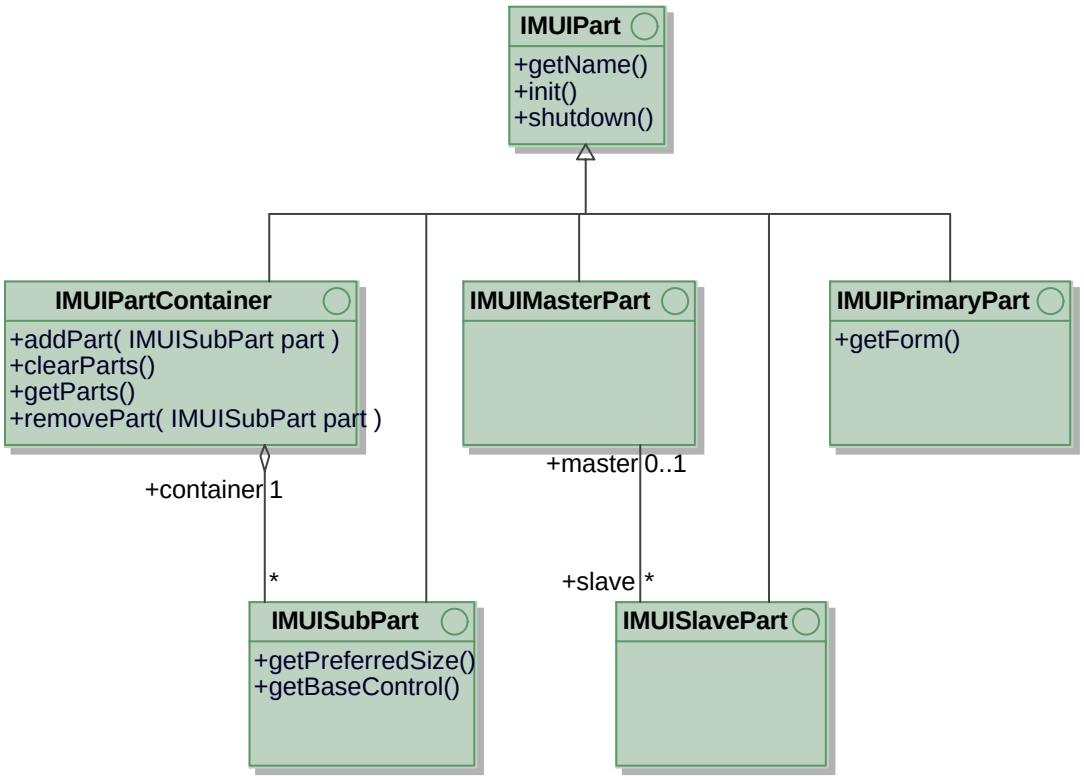
Käyttöliittymän ja käyttöliittymälogiikan toteuttavat itsenäiset Part-moduulit muodostavat ajonaikaisesti dynaamisen moduulihierarkian, joka kytkeytyy pohjatoteutuksen ohjaimeen (BaseController). Hallintakäyttöliittymän tarjoama käyttöliittymäkehys toteutettiin hyödyntämällä hallintakäyttöliittymän tarjoamaa Part-moduuleihin perustuvaa laajennusmekanismiä. Kuva 5.3 esittää hallintakäyttöliittymän tarjoaman sovelluskehyksen konkreettisen rakenteen, joka sisältää myös käyttöliittymäkehyksen. Kaikki kuvan esittämät laatikot, lukuun ottamatta alimmaista BaseController -laatikkoa, edustavat Part-moduulina toteutettua tai toteutettavaa komponenttia. Valkoisilla laatikoilla esitetään sovelluskehyksen ulkopuolelle jääviä varsinaisia sovelluskomponentteja, jotka voivat lisäksi muodostaa omia Part-moduulihierarkioitaan. Jokainen "..." -merkintä katkoviivoitettuine komponenttilaatikoineen kuvaa variaatiopistettä, jossa uusia komponentteja voi lisätä olemassa olevien rinnalle. Kuvassa on lisäksi merkittynä kerroshierarkian staattiset ja dynaamiset komponentit. Staattiset komponentit luodaan (BaseController palveluineen sekä ensisijainen Part-moduuli) aina hallintakäyttöliittymää käynnistettäessä. Dynaamiset komponentit (kaikki muut) voidaan luoda ja tuhota ajonaikaisesti tarpeen mukaan.



Kuva 5.3: Hallintakäyttöliittymän sovelluskehyksen konkreettinen rakenne ja variaatiopisteet

Tasan yhden Part-moduulin on toimittava hallintakäyttöliittymän ensisijaisena Part-moduulina ja ohjelman pääikkunana, joka tarjoaa Windows Forms-sovelluskehystä hyväksikäyttävän ohjelman pääohjelmasilmukan. Ensisijaisen Part-moduulin tuhoutuessa hallintakäyttöliittymä sammuu.

Jokainen hallintakäyttöliittymän Part-moduuli toteuttaa vähintään yhden kuvassa 5.4 esitetyistä rajapinnoista. Kuvassa on listattuna ainoastaan ymmärtämisen kannalta tärkeät rajapintojen metodit.



Kuva 5.4: Hallintakäyttöliittymän Part-moduulien rajapinnat

IMUIPart-rajapinta esittelee kaikille hallintakäyttöliittymän moduuleille yhteiset ominaisuudet. Jokaisella moduulilla on käyttöliittymässä näytettävä nimi. Nimen ei tarvitse olla yksilöllinen ja se saa muuttua suorituksen aikana. Tämä rajapinta esittelee myös moduulin käynnistyksen ja sammutuksen yhteydessä kutsuttavat metodit.

IMUIPartContainer-rajapinta on tarkoitettu toteutettavaksi sellaisille moduuleille, joiden on tarkoitus pitää sisällään IMUISubPart-rajapinnan toteuttavia alimoduuleita. IMUIPartContainer-rajapinnan toteuttava moduuli määrittelee alimoduulien esitystavan. IMUIPartContainer-rajapinta esittelee säiliön käsittelylle tyypilliset metodit alimoduuleiden lisäykseen, poistoon ja hakuun.

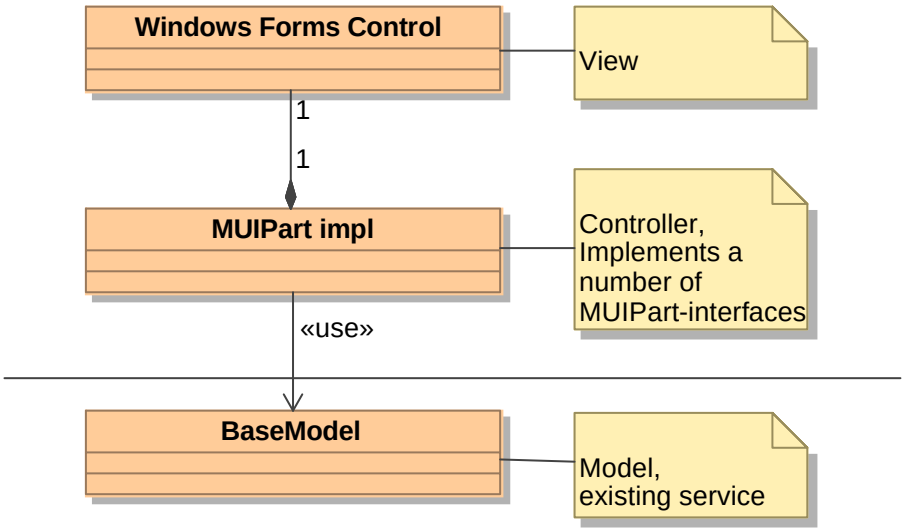
IMUISubPart-rajapinta on tarkoitettu alimoduuleille, joita voidaan säilöä IMUIPartContainer-rajapinnan toteuttamissa moduuleissa. IMUISubPart-rajapinta esittelee hakumetodit alimoduulin käyttöliittymäelementille ja tämän käyttöliittymäelementin suositellulle esityskoolle. IMUIPart-IMUIPartContainer-IMUISubPart-rajapintakolmikko ei muodosta Rekursiokooste-suunnittelumallin mukaista rakennetta, koska kaikki moduulitoteutukset eivät välttämättä tue alimoduulin roolissa olemista. Moduulien toteuttamien rajapintojen tarkastelu suoritetaan ajonaikaisesti.

IMUIMasterPart-rajapinnan toteuttavat moduulit voivat toimia isäntä-roolissa, isäntämoduuleina, IMUISlavePart-orjamoduuleille. Isäntämoduulilla on hallintaoikeus orjamoduuleihin, ja esimerkiksi isännän sammuessa se voi käskeä kaikkia hallitsemiaan orjamoduuleita sammumaan. Isäntämoduuli-orjamoduuli-suhteet ovat erillään moduulisäiliö-alimoduuli-suhteista, jolloin isäntämoduuli voi säiliöhierarkiassa olla samalla tasolla kuin tämän hallitsemat orjamoduulit.

IMUIPrimaryPart on tarkoitettu moduuleille, jotka pystyvät toimimaan Windows Forms –sovellusalustaan [MS10a] perustuvan sovelluksen käyttöliittymähierarkian alkupisteenä. Tällaisia ensisijaisia moduuleita voi olla ajossa kerrallaan yksin, ja se toimii samalla ohjelman pääsilmukkana. Rajapinta esittelee moduulin toteuttaman Windows Forms –ikkunan hakuun tarkoitetun metodin.

5.3.3. Hallintakäyttöliittymän moduulien suunnitteluperiaatteet

Moduulien sisäinen rakenne on vapaa, mutta myötäilee tyypillisesti MVC-arkkitehtuurityyliä. Kuvassa 5.4 on esitettynä luokkakaavio hallintakäyttöliittymän moduulitoteutuksen MVC-arkkitehtuurityylin mukaisesta rakenteesta.

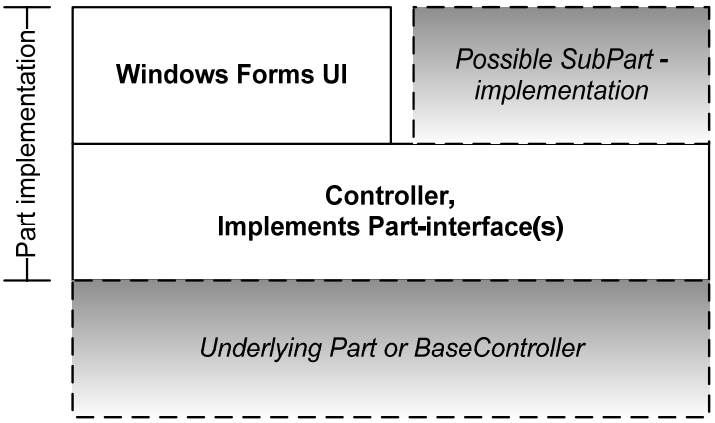


Kuva 5.5: Luokkakaavio Part-moduulin tyypillisestä MVC-arkkitehtuurin mukaisesta rakenteesta

Moduulitoteutukset eivät yleensä itse toteuta minkäänlaista tietomallia, vaan mallina käytetään pohjatoteutuksen tarjoamaa yleistä mallia (BaseModel). Yhteisen mallin käyttäminen parantaa toteutusten yhteensopivuutta ja korvattavuutta.

Ohjaimena toimii hallintakäyttöliittymän moduulirajapintoja toteuttava luokka, joka toteuttaa moduulin varsinaisen toiminnallisen logiikan. Ohjain sisältää lisäksi viittauksen Windows Forms –sovelluskehyksen mukaiseen käyttöliittymäelementtiin, joka toteuttaa moduulin varsinaisen käyttöliittymän ja käyttöliittymälogiikan.

Moduulin suhde muihin moduuleihin ja hallintakäyttöliittymän pohjatoteutukseen on esitelty kuvassa 5.5. Moduuleiden muodostama rakenne on hierarkkinen. Moduulin oman toteutuksen osuus on esitetty kuvassa valkoisilla laatikoilla.



Kuva 5.6: Part-moduulin rakenne kuvattuna kerroshierarkiana

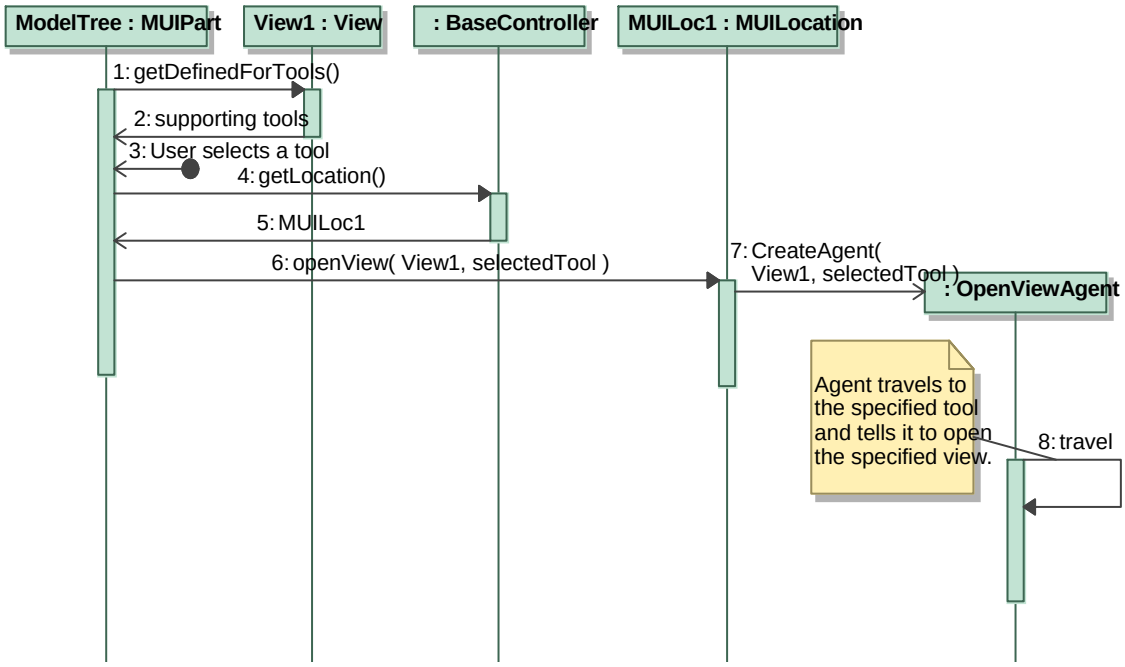
Alimoduulit liitetään niitä säilövän moduulin moduulirajapinnan toteuttavan ohjaintoteutuksen päälle, eikä moduulin näkymäkomponentti ole pakotettu olemaan mukana hallintakäyttöliittymän moduulirajapintatoteutusten mukaisessa hierarkiassa. Tämä ei kuitenkaan estä muodostamasta hierarkkisia käyttöliittymiä, ja käytännössä lähes aina alimoduulien käyttöliittymäelementit esitetäänkin niitä säilövien moduulien käyttöliittymäelementtien sisällä.

5.4. Työkalujen hallinta

Trinity-työkaluympäristön työkalujen hallinta hallintakäyttöliittymässä käyttää hyväkseen Trinity-työkaluympäristön tarjoamaa agentteihin perustuvaa integrointiarkkitehtuuria. Kirjoitushetkellä hallintakäyttöliittymässä hyödynnetään ainoastaan näkymien avaukseen sekä tietokannan synkronointikäskyihin liittyviä agentteja.

Seuraavassa esimerkissä kuvataan tapa, jolla hallintakäyttöliittymä ja sen moduulit voivat suorittaa työkalujen hallintaan liittyviä toimenpiteitä. Esimerkki liittyy kohdassa 4.3 määriteltyyn näkymän avaustoimintoon ja kuvaa näkymän avauksen kaksivaiheisen tapahtumaketjun.

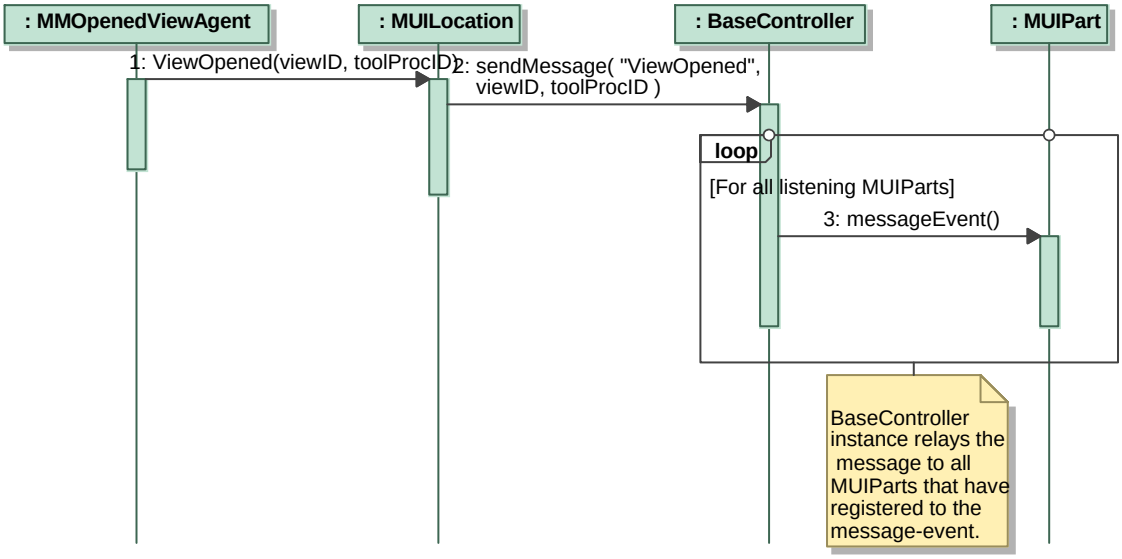
Kuva 5.7 esittää sekvenssikaavion näkymän avaukseen liittyvistä tapahtumista hallintakäyttöliittymässä. Alussa käyttäjä avaa näkymän avausvalikon (ks. kohta 4.3, Kuva 4.11, ”Open With...”) mallipuussa olevasta näkymää edustavasta solmusta. Avausvalikon avauksen yhteydessä valikon sisältö tuotetaan tietokannassa kuvattujen työkalujen näkymätyyppitukitietojen perusteella. Esimerkiksi näkymätyypille ”UML2_3_Classes_Visio_Class_Diagram”-näkymätyypille on määritelty sitä tukeviksi työkaluiksi yksi työkalu, ”Visio”, jolloin avausvalikkoon tuotetaan dynaamisesti valinta kaavion avaamiseksi Visio-työkalussa.



Kuva 5.7: Sekvenssikaavio näkymän avauksesta

Kun käyttäjä valitsee avausvalikosta haluamansa työkalun, mallipuumoduuli kysyy hallintakäyttöliittymän pohjaohjaimelta (BaseController) viitteen hallintakäyttöliittymän yhteiseen agenttiarkkitehtuurin Paikka-käsitteen mukaiseen paikkatoteutukseen (MUILocation). Mallipuukomponentti käskee tämän jälkeen paikkatoteutusta avaamaan em. näkymän käyttäjän valitsemassa työkalussa. Paikkatoteutus luo uuden ilmentymän näkymien avaamiseen erikoistuneesta agenttimäärittelystä, jonka jälkeen agentti lähtee välittömästi suorittamaan tehtäväänsä.

Työkaluilla on velvollisuus ilmoittaa hallintakäyttöliittymälle niiden avaamista näkymistä. Tämä vaihe näkymien avauksessa on tärkeä, koska työkaluilla on mahdollisuus avata näkymiä myös itse, ilman hallintakäyttöliittymältä tullutta käskyä. Hallintakäyttöliittymä on kuitenkin käyttäjän näkökulmasta paikka, jossa näkymien avausta ja sulkemista voidaan keskitetysti hallita, joten hallintakäyttöliittymän on oltava tietoinen kaikista avoimista näkymistä. Kuva 5.8 esittää sekvenssikaavion em. näkymän avauksen ilmoitusprosessin vaiheista.



Kuva 5.8: Sekvenssikaavio näkymän avautumisesta saapuvasta viestistä

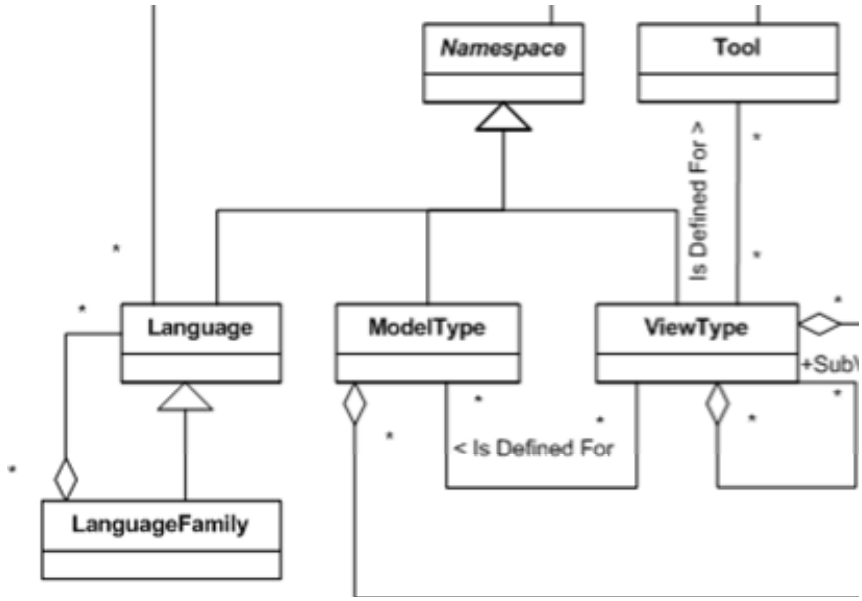
Kun työkalu on saanut avattua näkymän onnistuneesti, se lähettää näkymien avausilmoituksiin erikoistuneen agentin ilmoittamaan tapahtumasta hallintakäyttöliittymälle. Tämä agentti siirtyy hallintakäyttöliittymän paikkailmentymään ja tekee ilmoituksen avatusta näkymästä ja siitä, missä työkalussa näkymä avattiin (Kuva 5.8, kohta 1). Hallintakäyttöliittymän paikkailmentymä välittää tiedon eteenpäin hallintakäyttöliittymän pohjaohjaimelle (Kuva 5.8, kohta 2), joka edelleen välittää tiedon hallintakäyttöliittymän yhteiseen tiedonantokanavaan. Jokainen tiedonantokanavaa kuunteleva moduuli saa tiedon avatusta näkymästä (Kuva 5.8, kohta 3). Moduulit voivat ilmoituksen saatuaan tarvittaessa ilmaista tiedon avatusta näkymästä käyttäjälle.

5.5. Moduulitoteutuksien yksityiskohtia

Hallintakäyttöliittymässä kaikki käyttäjälle esitetyt käyttöliittymät ja niihin liittyvä toimintalogiikka toteutetaan omina moduuleinaan. Seuraavissa aliluvuissa on esiteltynä tämän diplomityön kannalta oleellisten moduulitoteutuksien yksityiskohtia.

5.5.1. Mallipuumoduuli

Mallipuun solmujen toiminnot tuotetaan dynaamisesti tietokantaan tallennettujen metametamalliin perustuvien määrittelyjen mukaisesti. Mallipuun toimintoihin liittyvä osa Trinityn metametamallista on esitettynä kuvassa 5.9.



Kuva 5.9: Otos Trinityn metametamallista (katso liite 1)

Järjestelmän tukemat mallityypit on määritelty tietokannassa mallityyppimäärittelyinä (Kuva 5.9, ModelType). Näistä tyypeistä ainoastaan konkreettisiksi määritellyt mallityypit esitetään käyttöliittymässä järjestelmän tukemina mallityyppeinä, joiden mukaisia mallityyppejä tietokantaan voi luoda.

Näkymät luodaan aina johonkin malliin. Mallin tukemat näkymätyypit päätellään mallin mallityypin ja näkymätyyppien välisistä määrittelysuhteista (Kuva 5.9, ModelType- ja ViewType-luokkien välissä suhde ”< Is Defined For”). Suhteet ilmaisevat onko tietty näkymätyyppi määritelty tietylle mallityypille, eli näkymän luonnin tapauksessa yksinkertaisesti, voidaanko tietyn näkymätyypin tyyppisiä näkymiä luoda valittuna olevan mallin mallityypin mukaiseen malliin.

Työkalutuki näkymätyyppien ja työkalujen välillä on määritelty vastaavanlaisilla suhteilla (Kuva 5.9, ViewType- ja Tool-luokkien välinen ”Is Defined For >”-suhde). Mallipuussa näkymän avausvalinnat tuotetaan dynaamisesti näiden suhteiden perusteella.

Kohdassa 4.3 esitetyn määrittelyn mukaisesti mallit voivat muodostaa alimallihierarkioita ja näkymät alinäkymähierarkioita. Näitä koostesuhteita voidaan vapaasti muokata esitetyssä mallipuussa. Nämä koostesuhteet määritellään aina koostavaan malliin tai koostavan näkymän sisältämään malliin.

Näkymillä on olemassa aina tasan yksi malli, jossa näkymä sijaitsee. Tätä suhdetta kuvaa näkymän luonnin yhteydessä automaattisesti malliin luotava suhde, joka osoittaa näkymän varsinaisen sijainnin. Tätä suhdetta ei saa poistaa, jotta tieto näkymän varsinaisesta sijainnista ei katoaisi.

Näkymän voi linkittää myös muihin malleihin kuin sen varsinaiseen säilövään malliin, esimerkiksi kun halutaan koostaa johonkin malliin viittauksia muissa malleissa sijaitseviin näkymiin. Näitä suhteita voi lisätä ja poistaa vapaasti ja ne säilötään aina siinä mallissa, josta halutaan tehdä viittaus näkymään.

Edellä mainituista suhteista voi muodostua päättymättömiä viittausketjuja, joiden johdosta mallipuun tuottamisessa rekursiivisesti ei voida luottaa siihen, että mallipuun koko olisi rajallinen. Tästä johtuen esitettävän mallipuun alisolmuja ladataan ”laiskasti” sitä mukaa, kun niiden isäsolmuja avataan.

5.5.2. Kategorisointimoduuli

Kohdan 4.4.1 mukaisesti kategoriat ovat pohjimmiltaan tyypillä laajennettuja avainsanoja. Avainsanat ja niihin liittyvät tyypit sekä suhteet säilötään mallissa, joka perustuu niiden kuvaamiseen erikoistuneeseen mallinnuskieleen.

Hallintakäyttöliittymässä kaikki kategorisointitieto säilötään yhteen malliin. Tähän sisältyvät avainsanaelementit, jotka edustavat kategorioita, kategoriatyypit, sekä kategorioihin liittyvät suhteet. Näitä suhteita ovat pysyvät suhteet kategorian ja kategoriatyypin välillä, sekä suhteet kategoriasta kategorisoitavaan elementtiin tietokannassa, kuten malliin. Kategorisoitavat mallit eivät siis ylläpidä kategorisointitietoa itsestään. Erillisen kategorisointitietoon erikoistuneen mallin lähestymistavassa on hyvänä puolena etenkin se, että kategorisointitiedon ja kategorisoitavan tiedon välille ei synny ylimääräisiä riippuvuuksia. Erillinen kategorisointimalli voidaan tarvittaessa vaihtaa, jolloin järjestelmää voidaan tutkia täysin erilaisesta näkökulmasta.

6. ARVIOINTI

Tässä luvussa käydään läpi työssä esitettyjen ratkaisuiden vastaavuus luvussa 3 esiteltyihin vaatimuksiin ja pohditaan valittujen ratkaisujen hyviä ja huonoja puolia. Lopuksi esitellään tämän diplomityön aiheeseen liittyviä julkaisuja.

6.1. Vaatimuskattavuusmatriisit

Seuraavissa taulukoissa arvioidaan työssä esitettyjen ratkaisuiden vaatimuskattavuutta. Kattavuutta kuvataan arvoilla {0, %, 1}, jossa 0 kuvaa täysin kattamatonta, % keskinkertaisesti katettua ja 1 hyvin katettua vaatimusta. Taulukko 5 kartoittaa työssä esiteltyjen ratkaisuiden ja toteutuksen kattavuuden mallien ja näkymien hallintaan liittyviin vaatimuksiin. Kokonaisuutena mallien ja näkymien hallinnan vaatimukset katettiin hyvin.

Taulukko 5: Mallien ja näkymien hallinnan vaatimuskattavuusmatriisi

Vaatimus	Kattavuus	Selitys
V01. Perustietojen esitys	1	
V02. Perustoiminnot	1	
V03. Näkymien ja mallien hierarkkisuus	1	
V04. Suhteiden roolit	1	
V05. Näkökulmat	1	
V06. Näkökulmien joustavuus	1	
V07. Esitysjärjestys	%	Mallipuun ylimmällä tasolla solmut esitetään aina nimen mukaan aakkosjärjestyksessä. Tämä esitysjärjestys ei ole käyttäjän muokattavissa.
V08. Suhteiden muokkaus	1	
V09. Mallien ja näkymien raahaus toiseen työkaluun	1	

Taulukko 6 kartoittaa työssä esiteltyjen ratkaisuiden ja toteutuksen kattavuuden työkalujen hallintaan ja järjestelmän tilaindikaattoreihin liittyviin vaatimuksiin. Kokonaisuutena työkalujen hallintaan liittyvät harvat vaatimukset katettiin kohtuullisesti, mutta järjestelmän tilaindikaattoreita esitetään niukasti. Lisäksi kirjoitushetkellä ympäristön kehitysasteen vuoksi käyttäjäinformaation esittäminen jätettiin pois hallintakäyttöliittymästä.

Taulukko 6: Työkalujen hallinnan ja järjestelmän tilaindikaattorien vaatimuskattavuusmatriisi

Vaatimus	Kattavuus	Selitys
V10. Näkymien avaus	1	
V11. Työkalun valinta näkymää avattaessa	1	
V12. Tietokantayhteyden tiedot	%	Ainoastaan tietokantayhteyden kohdeosoitteet ilmaistaan käyttäjälle. Käyttäjällä ei ole tietoa esimerkiksi tietokantayhteyden eheydestä.
V13. Avoimet näkymät	0	Kirjoitushetkellä avoimien näkymien tilaa ei esitetä, sillä luotettavaa tapaa avoimien näkymien tarkkailuun ei ole toteutettu.
V14. Kirjautuneen käyttäjän tiedot	0	Kirjoitushetkellä Trinity-työkaluympäristö ei tue käyttäjätilejä eikä kirjautumista, minkä vuoksi ominaisuutta ei tässä vaiheessa toteuteta hallintakäyttöliittymään.

Taulukko 7 kartoittaa työssä esiteltyjen ratkaisuiden ja toteutuksen kattavuuden toimintojen keskittämiseen liittyviin vaatimuksiin. Jotkut näistä vaatimuksista, kuten V15, V16 ja V16, riippuvat hyvin paljon käyttäjän subjektiivisesta kokemuksesta. Tästä johtuen myös näihin vaatimukseen liittyviä kattavuusarviota tulee käsitellä suuntaa antavina subjektiivisina arvioina.

Taulukko 7: Toimintojen keskittämisen vaatimuskattavuusmatriisi

Vaatimus	Kattavuus	Selitys
V15. Korkea saatavuus	1	
V16. Häiritsemättömyys	1	
V17. Johdonmukainen käyttöliittymäkehys	1	
V18. Käyttöliittymäkokonaisuuksien ryhmittely	1	

Taulukko 8 kartoittaa työssä esiteltyjen ratkaisuiden ja toteutuksen kattavuuden muihin vaatimuksiin, jotka eivät liity mihinkään aikaisempaan vaatimuskategoriaan.

Vaatimukset V19 ja V20 ovat hyvin laajoja ja vaikeasti mitattavia, minkä vuoksi niiden kattavuusarviot ovat hyvin epätarkkoja.

Taulukko 8: Muiden vaatimusten vaatimuskattavuusmatriisi

Vaatimus	Kattavuus	Selitys
V19. Laajennettavuus	%	Hallintakäyttöliittymän moduulirakenne ei takaa yksittäisten moduulitoteutusten laajennettavuutta.
V20. Dynaamisuus	%	Kirjoitushetkellä toteutus ei tue avainsanamallin vaihtamista käyttöliittymästä.
V21. Mallinnustiedon välitön tallennus tietokantaan	1	
V22. Yhteiset palvelut	1	

6.2. Mallipuu

Tässä työssä esitelty mallipuumoduuli tarjoaa ratkaisun kaikkiin kohdassa 3.2 esitettyihin vaatimuksiin, lukuun ottamatta näkökulmiin liittyviä vaatimuksia (V05 ja V06). Käyttäjä kykenee suoriutumaan moduulin avulla Trinity-työkaluympäristön korkean tason mallinnustiedon hallintaan liittyvistä peruskäyttötapauksista moduulin tarjoamien toimintojen avulla.

Mallipuumoduuli tarjoaa vaatimusten mukaisten perusominaisuuksien lisäksi sekalaisia toimintoja erikoisempien operaatioiden, kuten baseline- ja template-kopiointien, suorittamiseksi. Nämä toiminnot jätettiin pois mallien ja näkymien hallinnan vaatimuksista, koska niiden ei katsottu sisältyvän vaadittaviin perusominaisuuksiin. Moduulin toteutuksessa pyrittiin kuitenkin ottamaan huomioon tulevaisuudessa ilmenevät vaatimukset uusille toiminnallisuuksille.

Tehokkuudeltaan mallipuumoduuli on riittävä tyypillisimmissä käyttötilanteissa, kuten luontioperaatioissa ja tiedon organisoinnissa. Tyypillisesti, kun käytössä on nopea verkkoyhteys keskitettyyn tietokantaan ja järjestelmässä on tuhansia malleja ja näkymiä, mallipuun päivittämiseen kuluu joitain sekunnin murto-osia.

Käytettävyydessä on vielä kuitenkin kehitettävää. Nykyisessä toteutuksessa mallipuussa voi olla vain yksi solmu valittuna yhtenä ajanhetkenä. Useamman solmun valinnan mahdollistaminen saattaisi parantaa käytettävyyttä etenkin malleja ja näkymiä järjesteltäessä ja raahattaessa niitä hallintakäyttöliittymän ulkopuolelle muihin työkaluihin. Toinen käytettävyyteen liittyvä parannuskohde on mallipuun esityksen selkeys. Nykyisessä toteutuksessa käyttöliittymä muuttuu helposti sekavaksi puun solmujen täyteläisen esitystavan takia. Nykyisessä puuhierarkiassa solmukohtaisen tilainformaation esitys on vaikeaa. Tulevaisuudessa nykyinen yksinkertainen puunäkymä muutetaan hierarkkiseksi taulukkonäkymäksi, jossa voidaan esittää jokaista solua varten enemmän ja selkeästi tietoa eri sarakkeissa.

6.3. Kategorisointi

Tiedon organisointitarpeita varten ja korkean tason mallinnustiedon näkökulmaratkaisuksi esitelty laajennettuihin avainsanoihin perustuva kategorisointimekanismi vastaa suoraan korkean tason tiedon hallinnan näkökulmien vaatimuksiin (V05 ja V06). Korkean tason mallinnustiedon organisointi kategorioiden avulla on todettu toimivaksi ratkaisuksi rajoittaa esitettäviä malleja ja näkymiä eri käyttäjäryhmien tarpeiden mukaisesti. Kategorisointimekanismi mahdollistaa hyvin joustavan tavan organisoida malleja ja näkymiä erilaisten käyttötarpeiden mukaisesti. Joustavuus tuo kuitenkin mukanaan jonkin verran kategorioiden hallintaan liittyvää työtä. Tätä taakkaa on kuitenkin pyritty keventämään liittämällä automaattisesti joitain useasti käytettyjä kategorioita malleihin luonnin yhteydessä, kuten mallin tyyppiä ja nimeä vastaavat kategoriat.

Kategoriaverkon projisointi kategoriapuuksi käyttäjän määrittelemien näkökulmien avulla on välttämätöntä, sillä kategorioiden lukumäärän oletetaan kasvavan lineaarisesti suhteessa mallien ja näkymien lukumäärään. Käytännössä kategorioita voidaan arvioida olevan noin viisi jokaista mallia ja näkymää kohden. Toisaalta kategoriapuiden rakenteen määritteleviin näkökulmiinkin liittyy käyttäjän tekemää hallintatyötä. Käyttäjän on määriteltävä ensin näkökulma, jonka perusteella kategoriapuu voidaan tuottaa. Tämä työtaakka on kuitenkin mitätön verrattuna kategorioiden näkökulmien tuomaan hyötyyn.

Tulevaisuudessa kategorisointimekanismin tärkein kehityskohde on tehokkuus. Trinity-työkaluympäristön tietokannan tarjoama yleinen mallinnustiedon hallintamekanismi ei ole tarpeeksi tehokas suurien avainsanamäärien hallintaan ja niihin liittyvien hakujen suorittamiseen. Eräs todennäköinen ratkaisu avainsanojen tehokkuusongelmiin on poiketa ympäristön yleisestä tavasta käsitellä mallinnustietoa ja luoda erityisesti avainsanoihin liittyviä erikoistuneita ja tehokkaita käsittelytapoja suoraan tietokannan tasolle.

6.4. Työkalujen hallinta ja ympäristön tilaindikaattorit

Työkalujen hallinta on kirjoitushetkellä sekä määritykseltään että toteutukseltaan yksinkertainen. Vaatimukset rajoittuvat näkymien avaukseen mallipuusta Trinity-työkaluympäristön tukemiin työkaluihin sekä halutun työkalun valintaan (V10 ja V11). Hallintakäyttöliittymä vastaa kumpaankin vaatimukseen suoraan mallipuumoduulin tarjoamilla näkymän avaukseen liittyvillä toiminnoilla. Tulevaisuudessa työkalujen hallintaa voidaan laajentaa esimerkiksi automaattisella ja keskitetyllä asennuspalvelulla (vrt. esim. keskitetyt Linux-ohjelmistopakettisäilöt [Deb10]).

Hallintakäyttöliittymässä esitetään kirjoitushetkellä hyvin rajoittunut määrä ympäristön tilaindikaattoreita. Tietokannan kohdeosoitteet sekä hallintakäyttöliittymässä nykyhetkellä suorituksessa oleva tietokantaoperaatio indikoidaan hallintakäyttöliittymän sivupalkissa. Tulevaisuudessa tärkeitä kehityskohteita ovat erityisesti tietokantayhteyden sekä integraatioarkkitehtuurin tilaa kuvaavat indikaattorit, käyttäjätiedon esitys sekä näkymien avonaisuustieto.

6.5. Toimintojen keskittäminen ja käyttöliittymäratkaisut

Hallintakäyttöliittymän tarjoama sivupalkki päänäkymineen ja paneeleihin perustuva käyttöliittymäkehys yhdessä vastaavat hyvin kohdan 3.4 vaatimuksiin. Sivupalkki mahdollistaa pienellä, mutta aina esillä olevalla käyttöliittymällään sekä korkean saatavuuden että häiritsemättömyyden. Tämä ratkaisu on kuitenkin tehty kompromissina käytön nopeuden kustannuksella.

Päästäkseen käsiksi varsinaisiin hallintakäyttöliittymän tarjoamiin toiminnallisuuksiin käyttäjä joutuu tekemään yhden ylimääräisen napsautuksen hiirellä tuodakseen paneeleissa sijaitsevat käyttöliittymäkokonaisuudet esiin. Tämä hidaste on pieni hinta toiminnallisuuden keskittämisestä saatuihin hyötyihin verrattuna, mutta toimintojen keskittämisen tarvetta ja siitä aiheutuvia haittoja on silti syytä analysoida ennen päätöksen tekemistä.

6.6. Arkkitehtuuri ja laajennusmekanismi

Hallintakäyttöliittymän rakenne koostuu pohjatoteutuksesta ja tämän päälle dynaamisesti kiinnittyvistä hierarkkisista moduuleista. Moduulien riippuvuuksien rajoittuminen hallintakäyttöliittymän pohjatoteutukseen parantaa hallintakäyttöliittymän ylläpidettävyyttä. Toisaalta, koska moduulit riippuvat pohjatoteutuksesta ja sen tarjoamista palveluista, ei moduulitoteutuksia voida sellaisenaan käyttää muissa ympäristöissä. Eräs ratkaisu tähän ongelmaan on kääriä moduulin yleiskäyttöiset osat omaksi kokonaisuudekseen ja toteuttaa hallintakäyttöliittymän moduulirajapintoja varten sovitin.

Moduulien vapaa pääsy kaikkiin hallintakäyttöliittymän pohjatoteutuksen tarjoamiin palveluihin, jotka pitävät sisällään myös Trinity-työkaluympäristön tarjoamat palvelut tietokannan käsittelyyn ja työkalujen väliseen kommunikointiin, luo joustavuutta moduulien toteutusnäkökulmiin. Moduulien toiminnan rajoittamiseen ei ole määritelty mitään mekanismia, minkä vuoksi esimerkiksi pahantahtoinen tai tietoturvan kannalta huonosti suunniteltu moduulitoteutus voisi aiheuttaa tuhoa järjestelmässä.

6.7. Aiheeseen liittyviä järjestelmiä

Bang ja muut esittelevät julkaisussaan [Ban10] hajautetun ohjelmistomallinnukseen tarkoitetun kehyksen nimeltään CoDesign. CoDesignin ydinominaisuuksia ovat reaaliaikainen mallien synkronointi maantieteellisesti hajautuneiden arkkitehtien välillä sekä mallinnustiedon ristiriitojen hallinta markkinoilta löytyvien ristiriitojen havaitsemiseen tarkoitettujen moottorien avulla. Näistä erityisesti hajautetun tiedon synkronointi sivuaa Trinity-työkaluympäristöä ja sen tiedon integraatioon liittyviä asioita. Maantieteellisesti hajautuneiden tietokantapalvelinten keskinäisten ristiriitojen hallinta muodostunee tulevaisuudessa merkittäväksi aiheeksi myös Trinityssä. Hallintakäyttöliittymä ei ota kirjoitushetkellä kantaa mallinnustiedon hajautuneisuuteen tai siinä mahdollisesti piileviin konflikteihin. Nämä ovat kuitenkin selkeitä hallintaa vaativia aspekteja, jotka voitaisiin tulevaisuudessa huomioida myös hallintakäyttöliittymässä.

Magic Draw –mallinnustyökalussa [Mag11] loogista mallinnustietoa ja kaavioita säilötään projekteissa, jotka säilötään tiedostoissa. Kaavioita hallitaan pääasiallisesti

projektin kaaviotyyppikohtaisissa kaaviolistoissa. Kaaviot ovat myös osa mallinnustietoa ja ne ovat näkyvissä mallinnustietoa puurakenteena esittävässä puuesityksessä. Tämä poikkeaa kuitenkin huomattavasti hallintakäyttöliittymän tavasta esittää malleja ja näkymiä. Hallintakäyttöliittymässä pääasiallinen esitysmuoto on puurakenne, joka sisältää ainoastaan malleja ja näkymiä, jättäen pois kaiken korkean tason hallintaan liittymättömän tiedon. Ylimääräinen ja turha tieto haittaa halutun tehtävän suorittamista. Hallintakäyttöliittymän tarjoama korkean tason mallinnustiedon hallintatapa joustavan organisointimekanismin (kategorisointi) kanssa skaalautuu Magic Drawin käyttämää tiedostopohjaista hallintatapaa paremmin. Tästä on erityisesti hyötyä suurissa organisaatioissa, sekä suuria ja monimutkaisia järjestelmiä mallinnettaessa. Tiedostopohjainen hallintatapa on kuitenkin pienessä mittakaavassa helpompi ja nopeampi käyttää.

7. YHTEENVETO

Tässä työssä toteutettiin Trinity-työkaluympäristöön integroitu modulaarinen hallintatyökalu, joka tarjoaa ympäristössä paikan yhteisten toimintojen keskittämiselle. Työkalu tarjoaa moduuleille ja moduulikehittäjille yhteisen sivupalkkiin ja paneeleihin perustuvan käyttöliittymäkehyksen sekä helpon pääsyn työkaluympäristön tarjoamiin palveluihin. Työkaluun toteutettiin ryhmä moduuleita, joka yhdessä tarjoavat korkean tason mallinnustiedon hallintaan ja hakuun erikoistuneen toimintokokonaisuuden. Korkean tason mallinnustiedon organisointia ja hakua varten suunniteltiin laajennettuihin avainsanoihin perustuva kategorisointimekanismi.

Toteutettu käyttöliittymäkehys luo selkeän ja helppokäyttöisen tavan hallita ajossa olevia moduuleita. Kehys on rakenteeltaan ja ominaisuuksiltaan yksinkertainen, mutta tarjoaa riittävän ja helppokäyttöisen tavan moduulitoteutusten käyttöliittymien esittämiseen hallintakäyttöliittymässä intuitiivisella tavalla. Aina näkyvissä oleva sivupalkki takaa helpon pääsyn moduulitoteutusten tarjoamiin toimintoihin.

Korkean tason mallinnustiedon hallintaan erikoistunut moduuli tarjoaa kokonaisuutena kohtuullisen helppokäyttöisen ja monipuolisen tavan hallita järjestelmän malleja ja näkymiä. Mallien ja näkymien hallinnan toimintatavassa ei kirjoitushetkellä ole kehitystarvetta. Suurimmat kehitystarpeet ovat varsinaisen toteutuksen käyttöliittymän yksityiskohdissa, kuten esityksen selkeydessä ja nykyisen tilan indikoinnissa.

Esitelty kategorisointimekanismi on erittäin joustava tapa organisoida tietoa. Sen avulla eri sidosryhmät kykenevät määrittelemään tarpeidensa mukaisia näkökulmia organisoitavaan tietoon. Mekanismi lisää kuitenkin käyttöliittymien monimutkaisuutta peruskäytössä ja saattaa olla jopa rasite yhden käyttäjän tapauksessa, jossa organisointitarpeet ovat vähäisiä. Kategorisointimekanismi toimii parhaiten monen sidosryhmän ja käyttäjän ympäristössä, jossa organisoitavaa tietoa on paljon ja organisointitarpeet monimuotoisia.

Tulevaisuudessa työkaluun tullaan todennäköisesti integroimaan uusia kokonaisuuksia, jotka antavat sille uusia rooleja järjestelmässä. Alustavasti suunniteltuja rooleja ovat muun muassa ympäristöön rekisteröityneiden käyttäjien hallinta, työnkulun hallinta sekä ympäristön konfiguraation hallinta.

Työn aihe liittyy monen käyttäjän yhteiseen hajautettuun mallinnustyökaluympäristöön ja sen sisältämän mallinnustiedon hallintaan. Tutkimusalueena aihe on suhteellisen epäkypsä, eikä aihetta sivuavia julkaisuja löydy kovinkaan helposti. Työn ratkaisujen merkitys tutkimuksen kannalta kiteytyy esiteltyjen korkean tason mallinnustiedon hallintaan liittyvien ratkaisuiden suunnitteluun ja toteutukseen. Toteutuksen avulla ratkaisut voitiin testata käytännössä ja todeta ne toimiviksi todellisessa käyttöympäristössä.

Työssä esitellystä kategorisointimekanismista saatetaan tehdä tulevaisuudessa tieteellinen julkaisu.

LÄHTEET

[Mag11] MagicDraw [WWW]. [Viitattu 29.9.2011]. Saatavissa: https://www.magicdraw.com.

[OMG10] Object Management Group, Unified Modeling Language, Infrastructure, version 2.3. [WWW]. [Viitattu 23.4.2011]. Saatavissa: http://www.omg.org/spec/UML/2.3/Infrastructure/PDF.

[EF11a] Eclipse Software Development Environment. [WWW]. [Viitattu: 10.6.2011]. Saatavissa: http://www.eclipse.org/

[Was90] Anthony I. Wasserman. Tool integration in software engineering environments. Lecture Notes in Computer Science, 1990, Volume 467/1990, pp. 137-149.

[Sha96] Shaw M. ja Garlan D. Software Architecture. Perspectives of an Emerging Discipline. 1996, Prentice Hall.

[Kos05] Koskimies K., Mikkonen T., Ohjelmistoarkkitehtuurit. 2005, Talentum.

[Muh11] Muhametsin S., Vartiala M., Peltonen J. A Model for Language-Independent Mobile Agents. In Proceedings of the 12th Symposium on Programming Languages and Software Tools. Institute of Cybernetics at Tallinn University of Technology. 2011. pp. 90-101.

[Pel10] Peltonen J., Felin M., Vartiala M. From a freeform graphics tool to a repository based modeling tool. In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume (ECSA '10). New York, NY, USA, 2010, ACM. pp. 277-284.

[EF11b] Eclipse Modeling Framework. [WWW]. [Viitattu: 10.6.2011]. Saatavissa: http://www.eclipse.org/modeling/emf/

[MS11a] Microsoft .NET Framework. [WWW]. [Viitattu 20.7.2011]. Saatavissa: http://www.microsoft.com/net/

[MS11b] The C# Language. [WWW]. [Viitattu 20.7.2011]. Saatavissa: http://msdn.microsoft.com/en-us/vcsharp/aa336809.aspx

[MS10a] Windows Forms. [WWW]. [Viitattu 20.7.2011]. Saatavissa: http://msdn.microsoft.com/en-us/library/dd30h2yb.aspx

[MS10b] Windows Presentation Foundation. [WWW]. [Viitattu 20.7.2011]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms754130.aspx

[MS11c] WPF and Windows Forms Interoperation. [WWW]. [Viitattu 20.7.2011]. Saatavissa: http://msdn.microsoft.com/en-us/library/ms751797.aspx

[Eug03] Eugster P.T., Felber P.A., Guerraoui R., Kermarrec A-M., The Many Faces of Publish/Subscribe, ACM Computing Surveys, Vol. 35, No. 2, June 2003, pp. 114-131.

[Sta73] Stachowiak, H. Allgemeine Modelltheorie. New York 1973. Springer.

[Kur07] Kurpjuweit, S., Winter, R. 2007. Viewpoint-based Meta Model Engineering. EMISA 2007 Enterprise Modelling and Information Systems Architectures. Proceedings of the 2nd International Workshop on Enterprise. Modelling and Information Systems Architectures, pp. 143-161.

[Deb10] The Debian GNU/Linux FAQ Chapter 7 - Basics of the Debian package management system. [WWW]. [Viitattu 7.8.2011]. Saatavissa: http://www.debian.org/doc/FAQ/ch-pkg_basics.en.html

[Ban10] Bang. J. y., Popescu D., Edwards G., Medvidovic N., Kulkarni N., Rama G. M., and Padmanabhuni S.. CoDesign: a highly extensible collaborative software modeling framework. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2 (ICSE '10), Vol. 2. New York, USA 2010. ACM. pp. 243-246.

LIITE 1: TRINITYN TIETOMALLIN METAMETAMALLI